8-17-2011 12:00 AM

Workflow-Net Based Cooperative Multi-Agent Systems

Yehia T. Kotb, *The University of Western Ontario*

Supervisor: Dr. Steven beauchemin, *The University of Western Ontario*
Joint Supervisor: Dr. John Barron, *The University of Western Ontario*
A thesis submitted in partial fulfillment of the requirements for the Doctor of Philosophy degree in Computer Science

WORKFLOW-NET BASED COOPERATIVE MULTI-AGENT SYSTEMS

(Spine title: Cooperative Multi-agent Systems)
(Thesis format: Monograph)

by

Yehia Kotb

Graduate Program in Computer Science

A Thesis submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy

The School of Graduate and Postdoctoral Studies
The University of Western Ontario
London, Ontario, Canada

THE UNIVERSITY OF WESTERN ONTARIO
School of Graduate and Postdoctoral Studies

CERTIFICATE OF EXAMINATION

Supervisor:

. . . .
Dr. Steven Beauchemin

Co-Supervisor:

. . . .
Dr. John Barron

Examiners:

. . . .
Dr. Xiaobu Yuan

. . . .
Dr. Ken McIsaac

. . . .
Dr. Marc Moreno Maza

. . . .
Dr. Mike Bauer

The Thesis by

Yehia Thabet Kotb

entitled:

Workflow-net based Cooperative Multi-agent Systems

is accepted in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy

. . . .
Date

. . . .
Chair of the Thesis Examination Board

Abstract

Workflow-nets are mathematical frameworks that are used to formally describe, model and implement workflows. First, we propose critical section workflow nets (abbreviated $\mathrm{WFCS}_{\mathrm{net}}$). This framework allows feedbacks in workflow systems while ensuring the soundness of the workflow. Feedback is generally not recommended in workflow systems as they threaten the soundness of the system. The proposed $\mathrm{WFCS}_{\mathrm{net}}$ allows safe feedback and limits the maximum number of activities per workflow as required. A Theorem for soundness of $\mathrm{WFCS}_{\mathrm{net}}$ is presented. Serializability, Separability, Quasi-liveness and CS-Properties of $\mathrm{WFCS}_{\mathrm{net}}$ are examined and some Theorems and Lemmas are proposed to mathematically formalize them. In this Thesis, we define some formal constructs that we then build upon. We define the smallest formal sub-workflow that we call a unit. We propose some mathematical characteristics for the unit and show how it can be used. We study similarities between units and whether two units can be used interchangeably or not. We then use composites out of simple units to build more complex constructs and we study their properties. We define the concept of cooperation and propose a mathematical definition of the concept. We discuss the concept of task coverage and how it affects cooperation. We claim that task coverage is necessary for any task to be achieved and therefore, a necessity for cooperation. We use mathematical methods to determine the task coverage and the candidate cooperative partners based on their capabilities that can contribute to the desired task. Workflow-net based cooperative behaviour among agents is proposed. First, we propose a cooperative algebra, which takes the desired objective of cooperation as a plan and then transforms this plan into a workflow-net structure describing dependencies and concurrency among sub-workflow elements constituting the overall plan. Our proposed cooperative algebra converts the plan into a set of matrices that model the cooperative workflow among agents. We then propose a cooperative framework with operators that assign tasks to agents based on their capabilities to achieve the required task.

Keywords: Workflow-net, Multi-Agent cooperation

Acknowledgement

I would like to thank my supervisors, Dr. Steven Beauchemin and Dr. John Barron, for their enormous help which lead to a successful completion of my Thesis. Drs. Beauchemin and Barron gave me countless hours to go over problems, were always extra helpful with whatever questions I had, and even provided me with extra financial aid to help my family and I survive.

I would like to thank my wife for her support and love. I would also like to thank my brother, parents, other members of my family and my friends who have helped me. Thanks also to other professors who have guided me during my graduate studies at Western.

Contents

List of Figures

Chapter 1

Introduction

Cooperative multi-agent systems is a crucial research topic in the field of robotics. Cooperative behavior is sometimes necessary for a multi-robot task, such as robot soccer. At other times it is beneficial for increasing performance. In this Thesis, we use an extension of Petri-nets called workflow-nets to solve the problem of cooperative multi-agent systems. Multi-robot systems are the case study for the proposed framework. We use workflow-nets to model the behavior of every agent involved in a cooperation process. Barron [13] used Petri-nets to model the synchronization communication of actions in an editor-referee system. The work presented in this Thesis is more encompassing than that as well as being provably theoretically sound.

1.1 Thesis Contributions

The following are the contributions of the Thesis:

1. We introduce a framework that synchronizes activities in mutually exclusive workflow-nets (critical sections),

2. We introduce a cooperative composite logic-based algebra for recursive workflow-net construction and

3. We introduce a workflow-net based cooperative framework for autonomous multi-agent cooperation.

1.2 Thesis Contents

This Thesis is organized as follows:

Chapter 2 surveys the current research on cooperative agents and workflow-nets.

Chapter 3 gives an introduction to Petri-nets and workflow-nets. In this Chapter, all the required concepts are presented and illustrated. These concepts are the bases for many proposed Theorems and Lemmas throughout this Thesis.

Chapter 4 introduces a new model for workflow-nets. This model deals with feedback and synchronization between cascaded activities in an automated operation. The proposed model is called workflow-nets with critical sections WFCS_{net}. It guarantees the soundness of a workflow-net while using feedback. A Theorem and Lemma of soundness are presented. Another Theorem shows the relationship between the soundness of a WFCS_{net} and the quasi-liveness and the CS-properties of that same net.

Separability and serializability are two features that are desirable in workflow-nets. Chapter 4 presents the two concepts and gives a Theorem that shows the soundness of a WFCS_{net} with serializability and separability.

Chapter 5 demonstrates a new composite logic based algebra for constructing a closed form algorithm, which can be used to build a workflow-net based on a given plan.

Chapter 6 proposes a cooperative framework using workflow-nets. The Chapter starts with a definition of a minimum cooperative object (we call this a unit). It also provides a mathematical description for what we call choice-independent units, similar units and identical units. We expand the definition to include sub-workflow-nets which are composed of several units (we call them compositions). We propose a Theorem of soundness for our cooperative framework. We then propose a cooperative operator. We then expand our platform to include N agents and we study its behavior when performing deterministic and non-deterministic time-sensitive tasks. Finally Chapter 7 ends the Thesis with conclusions and future work.

Chapter 2

Literature Survey

Robot-based problems such as robotic navigation often benefit from the advantages provided by multiple, cooperating mobile agents [16, 31, 41]. Such gains include improved performance and simplicity of robot design. In addition, there are common multi-agent tasks that cannot be carried out by a single robot, such as playing soccer and follow-the-leader swarms [8]. Conversely, predator-prey and terrain exploration problems are examples of tasks that can be performed by a single agent yet may be more efficiently addressed with multiple robots [9]. Cooperation among a group of robots is defined as the process of allocating and managing available resources to achieve a certain goal. Typically, these resources include time, actions, knowledge, sensor readings and computations. As such, a cooperative situation must satisfy various constraints on the goal, the tasks, and the robots themselves. These constraints can be summarized as:

1. Constraints on the nature and the amount of resources to be assigned to each robot and the time frame in which the goal must be reached,

2. Constraints on the tasks to perform, such as precedence ordering and the amount of time to complete tasks and

3. Constraints on task and robot synchronization.

Cooperating mobile agents must negotiate for resources and perform task planning and scheduling in order to accomplish common goals.

Aalst *et-al.* introduced the modeling of workflow systems with an extension to Petri-nets, known as workflow-nets [56, 54]. A workflow-net is a Petri-net with unique input and output places, and is said to be sound if it is possible for a token in the input place to reach the output place. Aalst also introduced constructs for token routing known as AND split and join, and OR split and join [1, 55].

Petri-net composition is an active area of research [62, 27, 65, 63, 48], where services may be composed on the basis of complementarity. For instance, send and receive services are complementary by nature and may be composed [10]. Most of the Petri-net based services are composed directly. Direct composition is the process of fusing two or more Petri-nets in a single one which provides a complete service. Strict conditions apply in direct Petri-net composition: for two communicating services A and B, the structure of messages received by A must match that of messages sent by B, and *vice-versa*. Services A and B must also consent to an exchange sequence for the composition to be logically correct [62]. In practice, direct composition is difficult because service designers cannot always anticipate the need for future composition.

Wei *et-al.* propose a web service composition methodology based on mediators which allow for the implementation of composition in the case of partial compatibility [62]. A mediator is an object that is built upon the services that need to be composed and works as an interface which unify their input and output message structures. The main objective of service composition using mediators is to fuse incompatible services.

Baldan *et-al.* and Aalst *et-al.* both introduce an extension to Petri-nets called *open Petri-nets* [10, 11, 57]. An open Petri-net is a workflow-net in which special sets of places act as service access points. The principle of soundness may not be respected in workflow-nets which possess multiple service access points. To overcome this problem, both Baldan *et-al.* and Aalst *et-al.* classified these access points into two types: the workflow places and the external places, acting as interfaces for the intermediate activity states [10, 57]. Two open Petri-nets can be composed by external places if and only if these types of places are complementary. This direction of research focuses on coupling services into a producer-consumer relationship.

Work distribution models have been investigated with the use of colored Petri-nets [49]. In a similar way, Aldred *et-al.* examine communication abstractions for distributed business processes [7]. These recent advances address process integration and distribution with various forms of Petri-net compositions.

Pederson proposed a technique to determine the minimum flow of composed Petri-nets given the minimal flows of its components [47]. The notion of minimum flow is addressed in the context of transition-based and place-based compositions. Determining the minimum flow is interesting in many applications as it represents the minimum processing required for an outcome.

Kindler *et al.* introduce a weaker condition for global soundness, first proposed by Aalst [33, 55]. According to Aalst, every sub-workflow must terminate with a token in its output place. Kindler *et al.* considers only the sub-workflows which have a token in their input places in determining the global soundness of the composed workflow.

Recently, Lima *et al.* [39] introduced various types of Petri-nets to model distinct views of the robotic task model, in addition to quantifying task performance and using learning techniques to improve the general efficiency of execution. However, soundness properties[1] were not addressed and it remains unclear whether his framework can guarantee successful cooperative goal completion. Zhang proposed a Petri-net framework for task-level planning [67] in which an algorithm that depends only on the goal and the constraints required to derive action sequences is proposed. Gerkey and Matari [26] presented a domain-independent framework for multi-robot task allocation in which it is shown that task-allocation may be thought of as an instance of the optimal assignment problem. Alternatively, Noborio and Edashige [45] proposed an on-line, deadlock-free path-planning algorithm for multiple agents operating in an infinite world. Sauro *et al.* [53] proposed a framework that defines the cooperation problem using three modules: the environment module, the action module, and the agent module. Both the environment and the actions are modeled with a labeled-transition system where the states represent the environment and the edges represent the various actions taken by the agent within the environment. Each pair of states is connected with a single edge to represent the possibility of the environment changing its state from state S to state S', when an action is performed. The agent is defined by the set of actions it can perform. In this case, the cooperation of two or more agents is the union of the two or more action sets that belong to the agents. The main disadvantage of this approach is that it cannot represent concurrency in a direct way. Moreover, when the environment is complex and the number of agents exceeds a certain limit, the framework becomes unsuitable for many applications, especially those that need to maintain

[1] If a Petri-net has an input token then it will eventually have an output token; i.e. it will execute and terminate.

real-time performance measures.

Herzig and Longin [32] developed a new logic of intention which addresses the problem of cooperation. They showed that cooperation not only involves the contribution of some agent to goal satisfaction but also the adoption of the beliefs and intentions of that agent so as to indirectly allow it to achieve its goal. They proposed various techniques for belief adoption and intention generation. However, they did not propose a solution for the case where two or more beliefs contradict each other.

Chaimowicz *et al.* [18] proposed a technique for tightly coupled multi-robot cooperation, in which robots work in concert to achieve a common goal. Their work focused on merging communication with simple control techniques in order to achieve cooperation. Their platform allows heterogeneous robots with different sensors and control mechanisms to cooperate in order to achieve complex goals. Their platform was tested using a follow-the-leader task.

Chong *et al.* [22] proposed a coordination control between telerobots in such a way as to overcome the problem of delayed round-trip signals. Their testing environment consisted of two robots and two human operators who remotely controlled the robots. Thus, the robots were not fully autonomous and their coordination approach was aimed at solving the delay problem only.

Botelho and Alami [15] proposed a decentralized multi-robot system scheme for loosely coupled task planning and negotiation. The main protocol consists of three services: task allocation, cooperative reaction and task execution. Task allocation is achieved by sending the same set of tasks to be executed to all robots, followed by a negotiation for task selection. Each robot creates a plan for achieving the task under consideration. Those robots unable to generate a plan move to an idle state, waiting for another task, whereas those who can generate plans are considered candidates. The best candidate is chosen by the robot asking for the service. A cooperative reaction occurs when a robot fails to achieve its task. The robot asks for help and evaluates the offers from other robots to choose the best one. If no other robot responds, the failed robot abandons the task. The execution task is responsible for synchronization between the set of cooperative robots and the execution of the required tasks. Together, these three services constitute a framework the authors call M+.

Alami *et al.* [4, 6] proposed a framework for multi-robot systems called the Martha Project. The Martha project deals with trans-shipment tasks using multi-robot systems (from 10 to 100 robots). The system is comprised of a central station and a set of autonomously guided vehicles that communicate among themselves and with the central station. They proposed a topological graph-based environment model that is suitable for the problem of robot cooperation. The topological graph models areas, routes, and docking stations. The model is known to each autonomous vehicle and when a task is submitted to a robot from the central station, it is required to refine the plan generated to achieve the task after coordinating the route access with the other robots. The coordination is achieved through a plan-merging paradigm that guarantees a conflict-free overall plan for the cooperating robots.

Lin *et al.* [40] proposed an agent-based robot control that achieves multi-robot cooperation for environment exploration. The proposed approach depends on merging the partial environment maps simultaneously obtained by many robots to constitute a complete mapping. Their experiments show good results when using two robots. However, the problem of coordinating more than two robots was not specifically addressed.

An interesting approach to robot cooperation was proposed by Yingying *et al.* [66]. They proposed a way to define a personality for each cooperating robot via a function with parameters that can be tuned to create different modes of personalities for the robots. The authors claim that giving the agents their own personality produces more cooperation modes. These modes can lead to better performance for the multi-agent system. They did not show how these modes can be controlled to yield a better performance.

Al-Jumaily and Kozak [3] proposed a robot cooperation framework that is based on negotiation. The negotiation among agents is achieved in several steps. The first step, performed by the agent who is required to attend to a task, is to negotiate with other agents, by broadcasting the negotiation request to all. In the meantime, all other agents listen to the broadcast. These steps (Broadcasting and Listening) achieve together the data sharing that is required for the agents to negotiate. The third step is for every agent to calculate its distance to the negotiator. After calculating all distances, a decision is made. The robot that is closest to the target is the one that will achieve the required goal.

Brokowski *et al.* [14] proposed a robot cooperation framework based on Finite State Automata (FSA). They designed cooperation protocols using FSA between two robots to manipulate and displace objects within a simple environment. The model they used did not address the problem of scalability. It is known that the complexity of representing concurrency with FSAs increases exponentially with the number of agents. To avoid this problem, they claimed that their framework is only applicable in simple environments.

Chan and Yow introduced a strategy-driven framework for multi-robot cooperation [20]. They used robot soccer as an application to demonstrate their framework. Each team has a set of strategies stored in a database in the form of a multi-dimensional cube. Each node of this cube is a single strategy and, according to the context, a team may change its strategy for a neighboring one within the cube. Their work represents a way to model a global plan for the whole team. However, cooperation between robots is not explicitly achieved; their work represents a role assignment based on a predefined set of global plans rather than explicit cooperation.

Chapter 3

Introduction To Petri-nets

Petri-nets were invented by Carl Adam Petri in 1939 for the purpose of describing chemical processes. He documented the Petri-net in 1962 as part of his dissertation, *Kommunikation mit Automaten* (communication with automata). Petri's work significantly advanced the fields of parallel and distributed computing and it helped define the modern studies of complex systems and workflow management.

A Petri-net is a directed bipartite graph with two types of nodes, namely **places** (circles) and **transitions** (solid rectangles). Transitions model discrete events that may occur. Places are pre- or post-conditions for the transitions they are connected to. Places and transitions are connected via directed weighted **arcs** and if these arcs are not weighted than the **weight** is assumed to be one. These integer weights determine the number of activities that flow from places along the arcs per transition. These activities are called **tokens** (small solid circles that reside by places). The distribution of tokens over places is called a **marking**. Figure 3.1 shows the structure of a Petri-net.

Arcs connect places to transitions and transitions to places but they never connect two

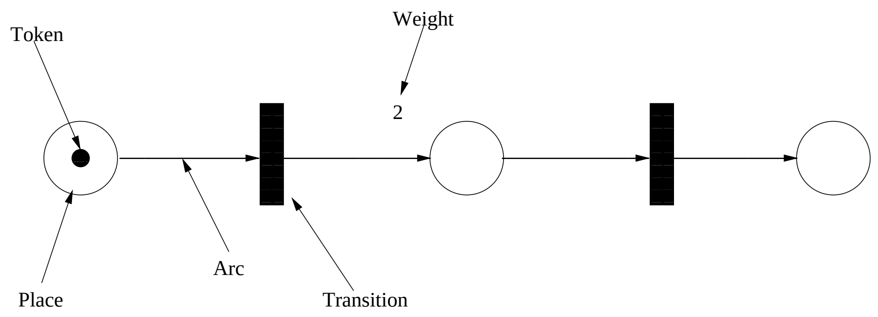


Figure 3.1: The structure of a Petri-net.

nodes of the same type together. When arcs run from places to a transition, these places are input places to this transition, and, when they run from a transition to places, these places are output places to this transition. A transition in a Petri-net is **enabled** if and only if there are tokens in all the input places to this transition and each input place contains a number of tokens that is greater than or equal to the weight of its connecting arc to this transition. After a transition is enabled, it will eventually **fire** by consuming tokens from its input places and producing tokens in its output places. The number of produced/consumed tokens per place is equal to the weight of the arc that connects this place with this transition. Figure 3.2 shows the firing process of a Petri-net. Figure 3.2(**a**) shows a petri-net with an initial marking of 4 tokens in place P_1. Transition T_1 is enabled because place P_1 has more tokens than the weight of the arc to T_1. In figure 3.2(**b**) , transition T_1 fires and 3 tokens are consumed from P_1 and 2 tokens are produced in place P_2 because the weight of the arc that goes from T_1 to P_2 is 2. Transition T_2 is then enabled. Consequently, in figure 3.2(**c**), transition T_2 fires and a token is consumed from P_2 and a token is produced in P_3. Therefore,in Figure 3.2(**d**), transition T_2 is still enabled and another token is consumed from P_2 and a new token is produced in P_3. Notice that transition T_1 is never enabled again during the scenario because the number of tokens in place P_1 is less than the weight of the arc that joins P_1 and T_1.

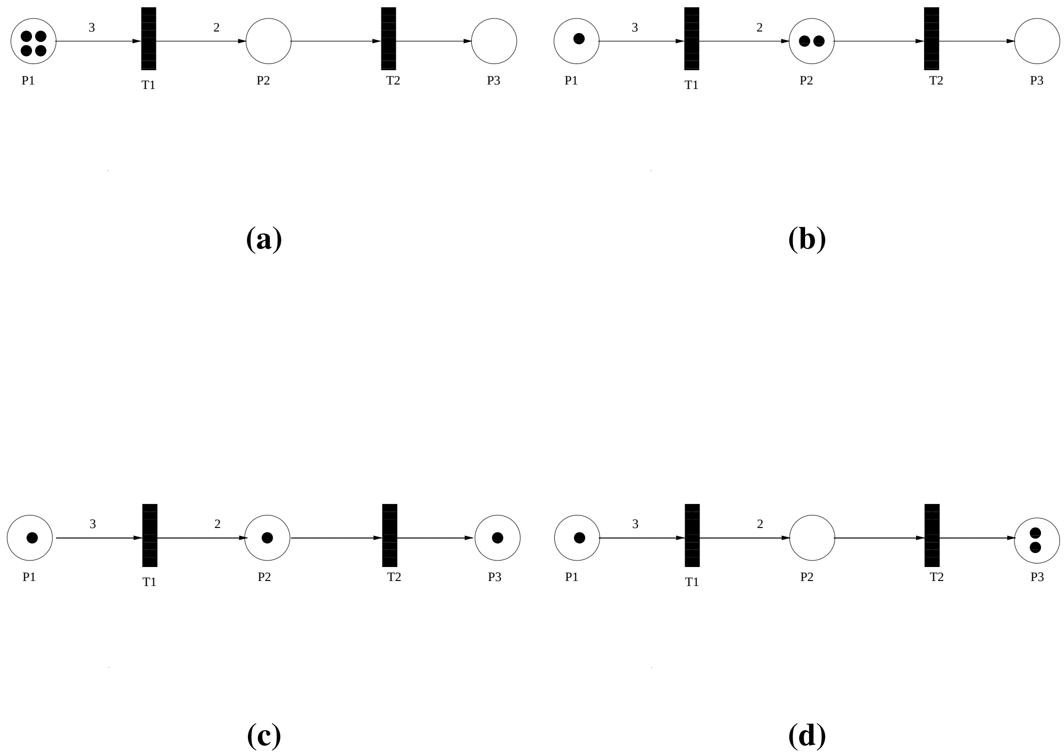


Figure 3.2: The firing of a Petri-net: **(a)** A petri-net with an initial marking of 4 tokens in place P_1. **(b)** Transition T_1 fires and 3 tokens are consumed from P_1 and 2 tokens are produced in place P_2. **(c)** Transition T_2 fires and a token is consumed from P_2 and a token is produced in P_3. **(d)** Transition T_2 is still enabled and another token is consumed from P_2 and a new token is produced in P_3.

Two transitions that have the same input place are called **choice-dependent** and in this case, the behavior of the Petri net is non-deterministic and one of the transitions is enabled and fired while the other has to wait to be re-enabled again. Figure 3.3 shows a Petri-net of two transitions that are not choice-free (choice-dependent).

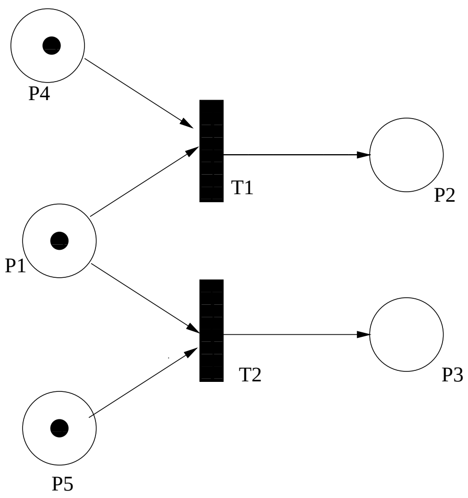


Figure 3.3: Transitions T_1 and T_2 are choice-dependent because if one of them fires, it will disable the other from firing. This is a non-deterministic case of Petri-nets, any one of them could fire before the other.

Consider a Petri-net for the famous dining philosopher's problem as shown in Figure 3.4. This problem illustrates a common multi-process synchronization problem first presented by Dijkstra and Hoare [61, 24]. The dining philosophers problem is summarized as five philosophers sitting at a circular table to either eat spaghetti or think. Eating and thinking actions in this problem are mutually exclusive and therefore any philosopher can either eat or think at any point of time. Between each pair of adjacent philosophers there is a fork that is placed and therefore, each philosopher has a fork on his left hand side and another one on his right hand side. The rule for eating in this problem is that two forks are needed for a philosopher to eat and a philosopher can only use the forks on his immediate left and right. In Figure 3.4,

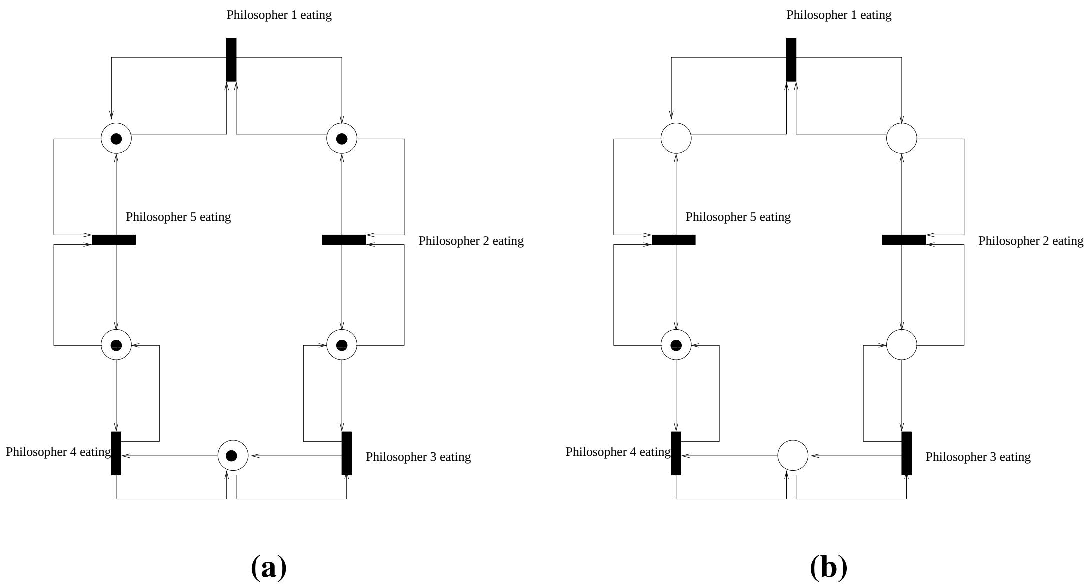


Figure 3.4: The Petri-net that models the dinning philosophers problem. Tokens in this case represent forks. **(a)** The initial state of the five philosopher model with forks on the right and left-hand sides of each philosopher. **(b)** The Petri-net marking when philosophers one and three eating. After they finish eating the transitions consume the tokens and the marking becomes like that shown in **(a)**.

forks are represented by tokens. Transitions represent the action of eating. When a transition is enabled and consequently fires (consumes two token from the two input place, one from each place) then the philosopher that is represented by this transition is eating. If the transition is enabled but did not fire yet or is not enabled at all then the philosopher represented by this transition is thinking.

Figure 3.5 shows some important features of Petri-nets. The first feature is concurrency

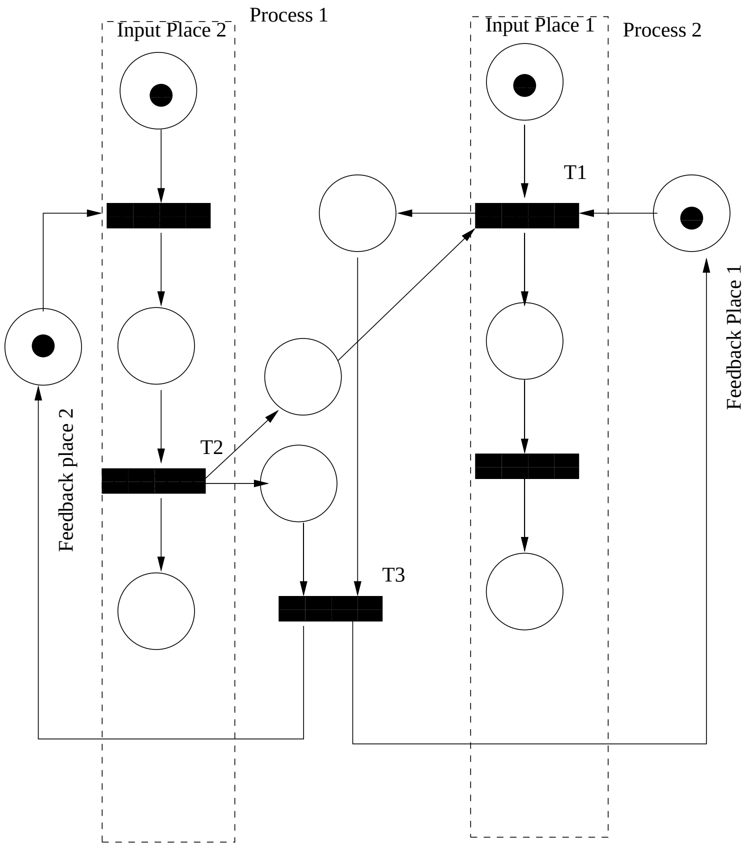


Figure 3.5: Two processes are modeled by a Petri-net. The two processes can not accept any new input unless both transitions T_1 and T_2 have been executed. Such synchronization is achieved by using transition T_3, which only becomes enabled when both T_1 and T_2 are fired. When T_3 is fired, it enables one or both transitions (T_1 and T_2) depending on which input places have tokens. Note that Process 2 can not deliver an output token unless Process 1 is executed. The 2 processes are delimited by the dashed lines.

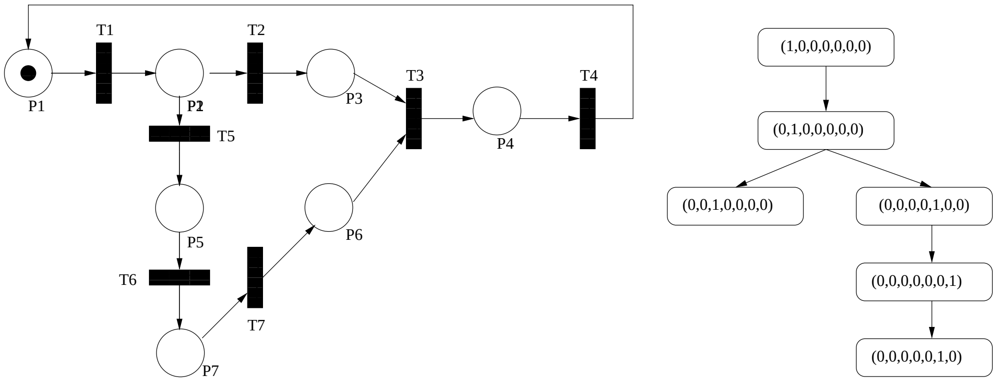
modeling. This Petri-net can model two processes working together. The second feature is synchronization. None of the processes can start a new execution instance unless the other process reaches a certain point in its execution, as defined by the model. The third feature is execution dependency. Note that Process 2 can not be executed until Process 1 is finished. Also note that the marking in the feedback places must be as shown for the very first execution.

3.1 Characteristics of Petri-nets

Characteristics of Petri-nets can be classified into **structural** and **behavioral** properties. Structural properties are those that describe how the Petri-net is built and how the topology describes the execution of tasks within the system. An example of these characteristics is boundness. Some higher level structural description for Petri-nets are state machine, marking graph, and siphons. Behavioral properties are those properties that describe how the machine will behave during run time. The behavioral properties depend on the structural properties and initial marking of the Petri-net. Examples for these properties are reachability, soundness, livness, serializability, separability and Controlled Siphons. In this Section, we describe some of the structural and bevabioral properties of the Petri-nets that is discussed through out this thesis.

3.1.1 Reachability

The **reachability** of a Petri-net depends on whether a certain marking can be obtained in the Petri-net from an initial marking. To study such a feature of Petri-nets, a reachability graph is constructed to determine all possible marking in this net, given some initial marking. Examples of reachability are shown in Figures 3.6 and 3.7. As shown in the Figure 3.6, T_3 needs tokens in both P_3 and P_6 to be enabled and consequently fired. T_2 and T_5 are competing for the token that will eventually reside in P_2 because they are choice-dependent. This token will eventually



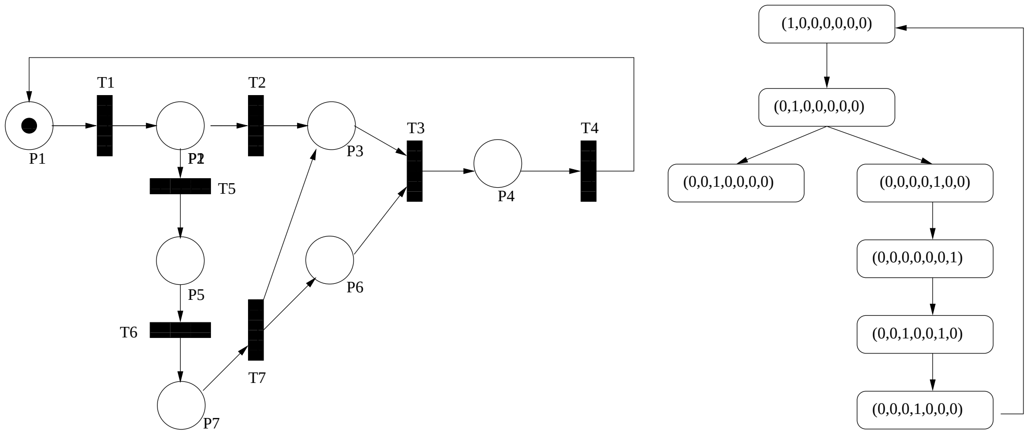
(a) **(b)**

Figure 3.6: Non-reachable markings in a Petri-net. (**a**): A Petri-net with an output place that is not reachable and (**b**): A reachability graph for the Petri Net shown in (**a**).

reside in either P_3 or P_6, depending on whether T_2 or T_5 will fire, and therefore, T_3 will never be enabled with this marking, and therefore, P_4 will never be marked. Figure 3.7, on the other hand, shows a case of reachability. The output of this Petri-net is reachable provided that transition T_5 consumes the token in place P_2 before transition T_2 does. The Petri-net in Figure 3.8 is **strongly connected** in that every node in the Petri-net is reachable from any other node in this Petri-net and therefore this net represents full reachability.

3.1.2 Livness

The **livness** of a Petri-net $\aleph$ is a property that indicates whether all the transitions that belong to $\aleph$ can be enabled or not. A Petri-net with fully reachable transitions is an example of liveness. Figure 3.6 shows a Petri-net that is not live because there are transitions that will never be



(a) **(b)**

Figure 3.7: Reachable markings in a Petri Net. (**a**): A Petri Net with an output place that is reachable and (**b**): A reachability graph for the Petri Net shown in (**a**).

enabled or fired. Figure 3.8, on the other hand, shows a live Petri-net because all transitions can be enabled.

3.1.3 Boundness

Boundness is a structural property of a Petri-net. It indicates how many tokens can reside in each place at any time. The maximum capacity for all places is the boundness of a Petri-net. If a Petri-net is 1-bound (the capacity for every place is a single token) then the Petri-net is said to be *safe*.

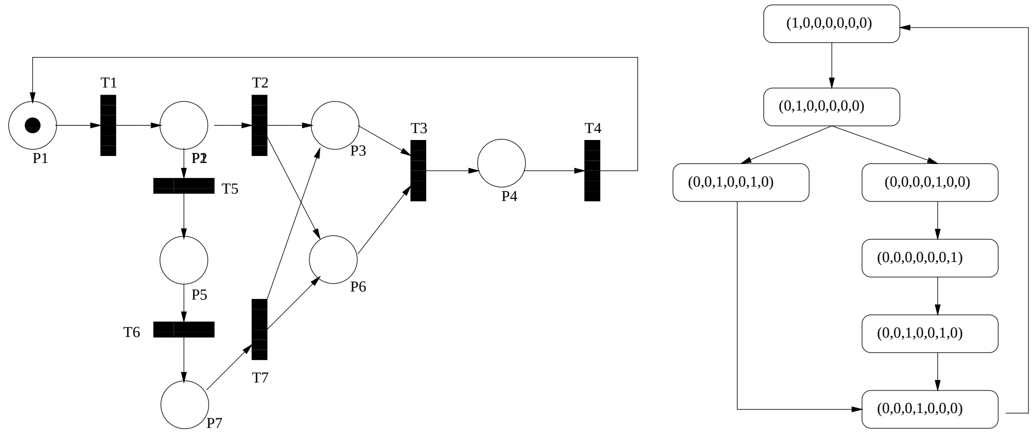


Figure 3.8: A strongly connected Petri-net. **(a)**: Every node in this Petri-net is reachable from any other node that belongs to this net. **(b)**: A reachability graph for the Petri Net shown in **(a)**. The graph shows that any marking can be reached from any other marking.

3.1.4 State Machine

A Petri-net is said to be a **state machine** if and only if for every transition there is a single input place and a single output place. Figure 3.9 **(a)** shows a Petri-net that is a state machine while Figure 3.9 **(b)** shows a Petri-net that is not a state machine (transition T_2 has 2 input places).

3.1.5 Marking Graph

A Petri-net is said to be a **marking graph** if and only if for every place there is a single input transition if any, and a single output transition if any. Figure 3.9 **(b)** is a marking graph. Figure 3.10 shows a Petri-net that is both a state machine and a marking graph at the same time.

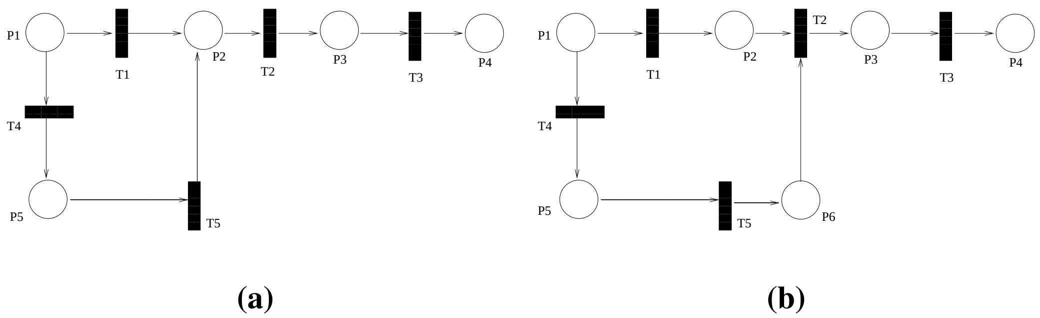


Figure 3.9: Different Petri-net structure nets. **(a)**: A State Machine Petri-net and **(b)**: A Marking Graph Petri-net.

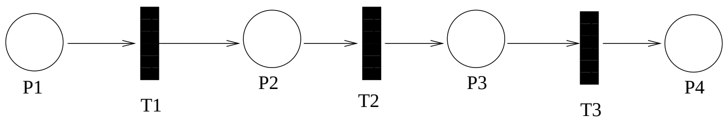


Figure 3.10: A Petri Net that is both a state transition and a marking graph.

3.2 Siphons

A Siphon is a place that is if insufficiently marked; it will never receive new tokens. An example of a siphon is place P_5 as shown in Figure 3.11.

3.3 Petri-net Symbols

The following Petri-net symbols occur most often in the literature and are used through out this thesis:

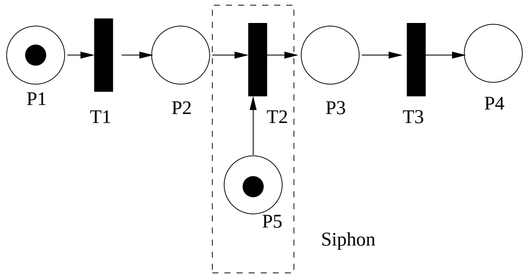


Figure 3.11: An example of a Siphon. When place P_5 loses its marking, it will never again be sufficiently marked.

- $\bullet t$ is the set of input places to transition t
- $t\bullet$ is the set of output places from transition t
- $\bullet p$ is the set of transitions that are connected to place p as input
- $p\bullet$ the set of transitions that are connected to place p as output
- $|N\rangle$ is the reachable set of nodes from node N
- $|\bullet t|$ is the number of input places to transition t
- $|t\bullet|$ is the number of output places to transition t
- P is the set of places
- T is the set of transitions
- p_k is the place number k
- t_k is the transition number k

- i is the set of input places of a Petri-net

- o is the set of output places of a Petri-net

- $M(k)$ or M_k is the Petri-net marking in step number k

- $M(0)$ is the initial marking of the Petri-net

- M_o is the output marking of the Petri-net

- $A \in |B\rangle$ means A is reachable from B.

Mathematically, a Petri-net is a 4-tuple:

$$\aleph = < P, T, F, W >, \tag{3.1}$$

where P is a set of places, T a set of transitions, F a set of arcs between transitions and places, expressed as $P \times T \cup T \times P$, and W is a vector containing the weights of the arcs in F.

3.4 Workflow-nets

Workflow management and business engineering processes consider industrial systems composed of business processes competing for resources [35]. The workflow space is spanned by three dimensions [35]. The 1^{st} dimension is the control flow dimension, which is concerned with the partial ordering of tasks. The 2^{nd} dimension is the resource dimension, in which resources are classified by identifying roles and organization units. The 3^{rd} dimension of a workflow is concerned with individual cases. Figure 3.12 shows the different dimensions of the workflow.

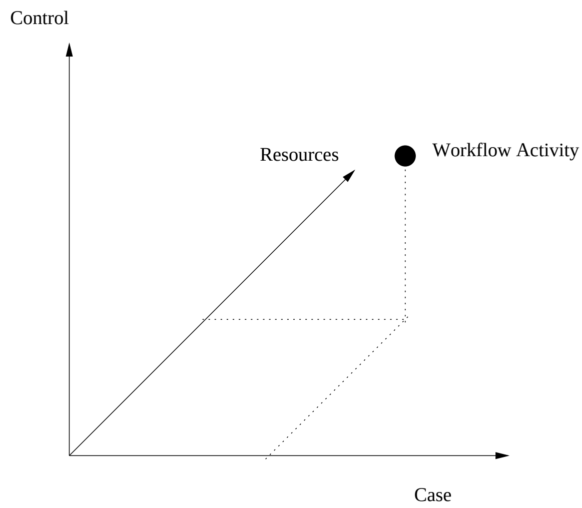


Figure 3.12: The three dimensions of a Workflow-net.

Workflow-nets ($\mathrm{WF}_{\mathrm{net}}$) are used to model the structural and dynamic behaviors of workflows. The structural behavior of a workflow defines task dependencies and their structure which guarantees the desired output. The dynamic behavior is how the structure reacts online with activities which are handled by the workflow. A workflow-net is a special type of Petri-net that has two special places, i and o, where i is the only place that does not have any input transitions and o is the only place that does not have any output transitions. Nodes i and o are called the **source** and **sink** nodes. Workflow-nets are preferred over normal Petri-nets in industrial applications because they guarantee the success of the industrial process (soundness). The following are definitions for Workflow-nets and their characteristics.

Definition 1 *A Petri-net $\aleph$ is a WF_{net} if and only if*

1. *$\aleph$ has an input place i, where $\bullet i = \phi$,*

2. *$\aleph$ has an output place o, where $o\bullet = \phi$ and*

3. *if a transition t^* is added to $\aleph$ such that $\bullet t^* = o$ and $t^*\bullet = i$, the Petri-net $\aleph^*$ becomes strongly connected.*

Note that t^* is a transition which connects the input to the output of the WF_{net}.

In the above definition, $\bullet i$ is the set of all input transitions to place i and $o\bullet$ is the set of output transitions from place o. When a Petri-net is strongly connected, there is a sound path between any two transitions in the net.

One of the Petri-net properties that is considered essential to workflow-nets is the property of soundness. **Soundness** guarantees that the Petri-net will eventually terminate and, at this moment, there will be tokens in the output place and all other places will be empty. This signifies that all activities will reach the output place and none of them will be "lost" inside the net. When automating a process, it is essential to make sure that the Workflow-net is sound.

Definition 2 *A WF_{net} $\aleph$ is sound if:*

1. $\forall\ M \in\ |M_i\rangle,\ \ M_o \in\ |M\rangle,$

2. $\forall\ M_k \in\ |M_i\rangle, if\ \ M_k \in\ |M_o\rangle$ *then* $M_k = M_o$ *and*

3. $\forall\ t \in T,\ \ \exists\ M \in\ |M_i\rangle,\ \ \ t \in\ |M\rangle$ *and* $M_o \in\ |t\rangle$

where M is a marking of the $\mathrm{WF_{net}}$, M_i is the input marking, M_o is the output marking, M_k is the marking at time k and t is a transition.

Other important properties of Petri-nets are **Safeness** and **Quasi-Liveness**. A Petri-net is safe if it is 1-bound; that is, each place has a maximum number of 1 of tokens. Quasi-Liveness means that $\forall t_i \in T$, t_i is firable in a finite time given marking M. In other words, all transitions will eventually be enabled and fired. If a transition T can not be enabled, then this transition is said to be **starving**. To guarantee the success and the feasibility of applying workflow-nets in industry the following three definitions are needed.

Definition 3 *The Controlled Siphons property (CS-property) states that a siphon is controlled, if and only if, for each reachable marking, the siphon remains sufficiently marked. A Siphon is a place that is, if insufficiently marked, will never again get new tokens.*

Figure 3.13 shows place P_5 as controlled siphon. This place is guaranteed to remain sufficiently marked through the firing of T_3.

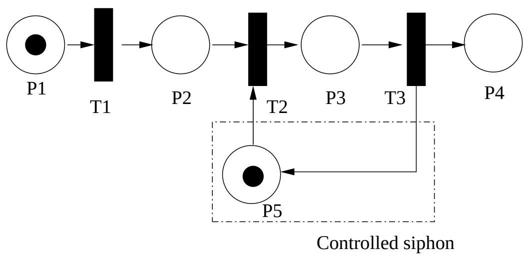


Figure 3.13: Controlled Siphon.

Definition 4 *Separability is a behavioral property which states that the behavior of a workflow-net with k tokens in the initial node is seen as a combination of the behavior of k copies of the net, each of them with one token in the initial node.*

Figure 3.14 shows a workflow-net that is not separable. Th two transitions T_1 and T_2 have to be enabled each by consuming one token from P_1 for the workflow to be able to terminate.

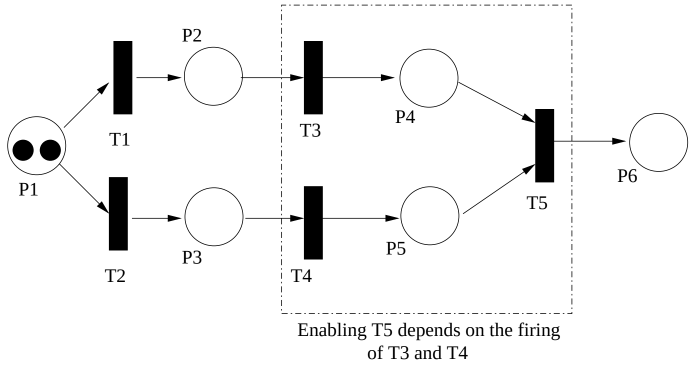


Figure 3.14: Non-Separable workflow-net.

So the two activities represented by the two tokens depend on each other for them to reach the output.

Definition 5 *Serialisability requires that the set of traces of a workflow-net with id-marking (in this case each token has an identifier) is equal to the set of traces of an abstraction of the workflow-net. In other words, an activity that exists in a workflow does not effect any other activity that co-exists in that same workflow.*

In other words, Serializability is the ability of seeing the execution of cascaded activities in a workflow-net as if each of them owns the workflow-net execlusively at any time.

Figure 3.15 shows a workflow-net that is not serializable. it is also not separable. A token needs to be consumed by T_3 in order to enable T_2. This means that the marking of one activity depends on the marking of the other.

Serialisability and Separability are related. Serialisability views the workflow as a pipeline while Separability views the workflow as a parallel system.

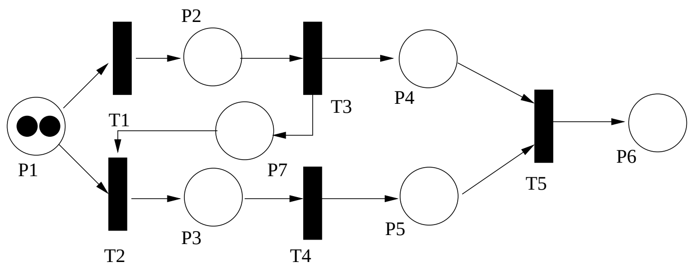


Figure 3.15: Non-Serializable workflow-net.

Chapter 4

Critical Section Workflow-nets

The concept of workflow management systems first appeared in the field of business process management [1]. Since then, workflow management has been applied to a wide spectrum of applications. Kotb *et al.* proposed a workflow management based health care operating system [35, 38, 37]. In later research, Kotb *et al.* proposed a workflow based cooperative platform for multi-robot systems [36]. Since there is a large diversity of scientific applications, each field presents its own view of workflow.

A Scientific Workflow Management System (SWMS) is a group of software modules that handles the modeling and the execution of a scientific experiment. It also models the dependencies among experiment activities and processes and manages resource allocation and utilization during the experiment [69]. In the last decade, this type of research has been done in the grid computing field and has made complex and intensive experimental processing feasible [68, 69].

4.1 Critical Sections and Workflow-nets

One of many things that makes applying workflow-nets on scientific applications infeasible is Workflow-net constraints. One of these constraints is that feedback and loops[1] are not permitted, as it threatens the soundness of the system [1]. Some workflow systems have critical sections, a set of tasks that can only be executed by one activity at a time. This type of workflow needs very careful handling so that the critical sections are maintained. Real time requirements are fulfilled if and only if the soundness property is guaranteed. We control a critical section in a workflow-net using a special feedback net. This net controls the flow of activity in the workflow-net while maintaining system soundness. We call this net a critical section workflow-net (denoted CS_{net}). This net is itself a workflow with special tokens known as control tokens. The original workflow-net and one or more of the control nets constitute what we call a workflow-net with critical sections (denoted WFCS_{net}). An example of a WFCS_{net} is shown in Figure 4.1. The following are definitions for our proposed WFCS_{net}:

Definition 1 *A critical section is a workflow-net or a sub-workflow-net that can not serve more than n activities at a time, where n is an integer, $1 \leq n \leq k$, and k is the maximum number of activities that can flow in the critical section at any point of time.*

Definition 2 *A CS_{net} Ψ is a control net for WF_{net} $\aleph$ if:*

1. *Ψ is a sound WF_{net},*

2. *i_Ψ is the input place of Ψ such that $\exists\ t_i \in \aleph$ where $i_\Psi \in t_i\bullet$,*

3. *o_Ψ is the output place of Ψ such that $\exists\ t_o \in \aleph$ where $o_\Psi \in \bullet t_o$ and*

[1]According to business process management, a workflow with loops is not a good design for workflow models.

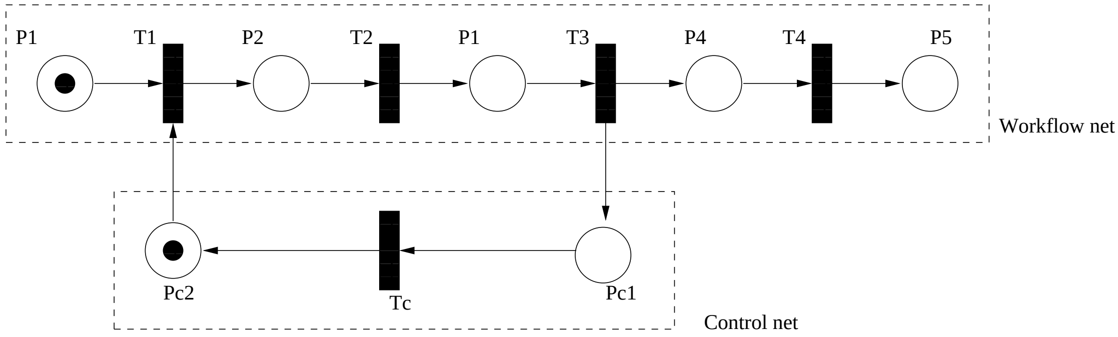


Figure 4.1: A critical section workflow-net denoted as $\mathrm{WFCS}_{\mathrm{net}}$. The $\mathrm{WF}_{\mathrm{net}}$ is defined by $\{P_1, T_1, P_2, T_2, P_3, T_3, P_4, T_4, P_5\}$. The input place is P_1 and the output place is P_5. The control net $\mathrm{CS}_{\mathrm{net}}$ is defined by $\{P_{c1}, T_c, P_{C2}\}$. Note that $M(P_{c2})$ is part of the initial marking, otherwise the net is not sound. After an activity is executed, P_{C2} must have the same inital marking. This is the minimum sufficient marking for P_{C2} as a controlled siphon. The initial marking of P_{C2} determines the number of activities that can flow in the controlled workflow-net at any point of time.

4. $\Psi \cap WF_{net} = \{i_\Psi \times t_i \cup t_o \times o_\Psi\}$ *where* $t_i \in WF_{net}$ *and* $t_o \in WF_{net}$ *and* $t_i \in |\ t_o\rangle$.

Definition 3 *A $WFCS_{net}$ can be described in terms of a WF_{net} and a CS_{net} as follows:*

$$WFCS_{net} = WF_{net} \bigcup CS_{net}. \tag{4.1}$$

Definition 4 *A $WFCS_{net}$ $\aleph$ is a 5-tuple:*

$$\aleph = \langle P \cup P_c, T \cup T_c, F, W \cup W_c, M(0) \cup M_c(0) \rangle\,, \tag{4.2}$$

where P is a set of resource token places in the $\mathrm{WF_{net}}$, P_c is a set of control places in the $\mathrm{CS_{net}}$, T is a set of transitions in the $\mathrm{WF_{net}}$, and T_c is a set of transitions in the $\mathrm{CS_{net}}$,

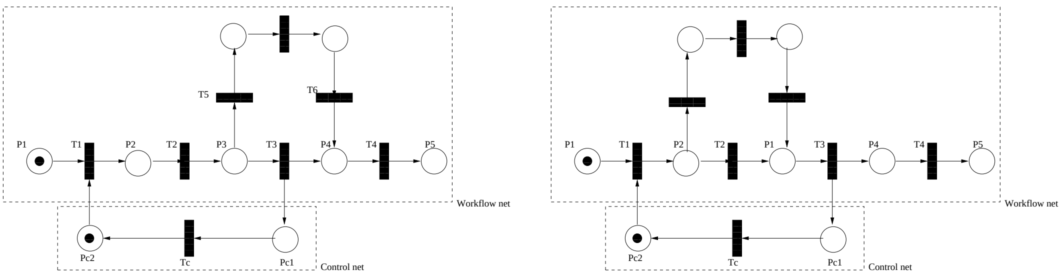
$$F = F_r \cup F_c \cup F_{cr},$$

$$F_r = T \times P \cup P \times T,$$

$$F_c = T_c \times P_c \cup P_c \times T_c,$$

$$F_{cr} = T \times P_c \cup P \times T_c \cup P_c \times T \cup T_c \times P,$$

W is the set of weights of the arcs of the $\mathrm{WF_{net}}$, W_c is the set of weights of arcs of the $\mathrm{CS_{net}}$, $M(0)$ is the initial marking of $\mathrm{WF_{net}}$, and $M_c(0)$ is the initial marking of $\mathrm{CS_{net}}$. The weights of F_{cr} and F_c are bound to 1. Figure 4.1 shows a simple $\mathrm{WF_{net}}$.



(a) **(b)**

Figure 4.2: $\mathrm{WFCS_{net}}$: Workflow-nets using control nets. **a)**: A $\mathrm{WFCS_{net}}$ that is not sound, as it violates the 4^{th} condition in Theorem 4.1.1 and hence, the control token is consumed by T_4 from place P_3 resulting in an insufficient marking in the siphon P_{c2}. **b)**: A $\mathrm{WFCS_{net}}$ that is sound.

Definition 5 *The following conditions must exist for a WF_{net} $\aleph = \aleph_f \cup \Psi$ to be a $WFCS_{net}$, where $\aleph_f$ is the original WF_{net} and Ψ is the CS_{net}:*

1. $\aleph_f = \langle P, T, F_r, W\rangle$ *is a sound* WF_{net}*,*
2. $\Psi = \langle P_c, T_c, F_c, W_c\rangle$ *is a sound* CS_{net}*,*
3. $P \neq \emptyset$ *and* $P_c \neq \emptyset$ *and* $P \cap P_c = \emptyset$*,*
4. $T \neq \emptyset$ *and* $T_c \neq \emptyset$ *and* $T \cap T_c = \emptyset$*,*
5. $\forall M_k \in [M(0)\rangle$ *and* $M_{ck} \in [M(0)\rangle$*, if* $M_k = M(0)$ *then* $M_{ck} = M_c(0)$ *and*
6. $\forall f \in F_{cr}$*,* f *is sound.*

As stated earlier, it is essential to guarantee soundness in workflow systems. In Figure 4.2, there are two models of $\mathrm{WFCS_{net}}$. The model on the left-hand side is a $\mathrm{WFCS_{net}}$ that is not sound, whereas the one on the right-hand side is sound. The following Theorem defines the soundness of the $\mathrm{WFCS_{net}}$.

In the following Theorem, we are adopting the following symbols:

- $\aleph$ is the controled $\mathrm{WF_{net}}$
- Ψ is the control net $\mathrm{CS_{net}}$ that is applied on $\aleph$
- $\aleph_f$ is $\mathrm{WFCS_{net}}$ where $\aleph_f = \aleph \cup \Psi$
- P_{ci} is the input place of Ψ
- P_{co} is the output place of Ψ

- i is the input place for $\aleph$
- o is the output place for $\aleph$
- t_i is a single transition that has P_{co} among its input places
- t_o is a single transition that has P_{ci} among its output places
- ϵ is a non-empty set of tokens
- $M(WF)$ is the marking of Workflow-net WF
- T is the set of transitions in $\aleph$
- P is the set of places in $\aleph$
- T_C is the set of transitions in Ψ
- P_C is the set of places in Ψ

Theorem 4.1.1 A WFCS$_{\text{net}}$ $\aleph_f$ is sound if and only if:

1. $\forall \ M_k$ such that $M_k \in \ | \ M(0) \cup M_c(0) \ \rangle, \ \ M_o \in \ | \ M_k \ \rangle$,
2. Ψ is a sound WF$_{\text{net}}$ with $M_c(0) = M_c(P_{co})$ and $M_c(0) > 0$,
3. $\aleph \cap \Psi = \{P_{co} \times t_i \cup t_o \times P_{ci}\}$ and
4. $\forall \ t_k \ \in \ | \ t_i \ \rangle, \ t_o \ \in \ | \ t_k \ \rangle$ or $\exists \ t_j \ | \ t_j = \bullet \bullet t_k$ and $t_o \notin \ | \bullet t_k \ \rangle$.

Proof 1. Since Ψ is a sound WF$_{\text{net}}$ with $M_c(0) = M_c(P_{co})$ and $M_c(0) > 0$, and $\aleph \cap \Psi = \{P_{co} \times t_i \cup t_o \times P_{ci}\}$,

2. therefore $\exists \ \epsilon \in \bullet t_i$.

3. Since $\forall \ t_k \ \in \ | \ t_i \ \rangle, \ t_o \ \in \ | \, t_k \ \rangle$ or $\exists \ t_j \ | \ t_j = \bullet \bullet t_k$ and $t_o \notin \ | \bullet t_k \rangle$,

4. therefore ϵ will not deviate to a non-sound path and will always be in a state that yields M_o.

5. Since $\forall \ M_k$ such that $M_k \in \ | \ M(0) \cup M_c(0) \ \rangle, \ M_o \in \ | \ M_k \rangle$,

6. therefore $\forall \epsilon \in \ M(0), \ \epsilon$ will eventually be in M_o,

7. therefore if the conditions of Theorem 4.1.1 are satisfied, then $\aleph_f$ is a sound $\mathrm{WF_{net}}$.

We now show the converse, namely that if the $\mathrm{WFCS_{net}}$ is sound, then these conditions are satisfied:

Proof 1. Since $\aleph_f$ is sound,

2. therefore tokens will always reach place o.

3. Since all tokens will eventually be part of M_o,

4. therefore $\forall \ M_k$ such that $M_k \in \ | \ M(0) \cup M_c(0) \ \rangle, \ M_o \in \ | \ M_k \rangle$, and $\exists \ t_i | P_{co} = \bullet t_i$ and

5. P_{co} is sufficiently marked.

6. Since P_{co} is sufficiently marked,

7. therefore Ψ is a sound $\mathrm{WF_{net}}$ and $\exists t_o \in \aleph | t_o = \bullet P_{ci}$.

8. therefore $\aleph \cap \Psi = \{P_{co} \times t_i \cup t_o \times P_{ci}\}$.

9. Since t_o is always reachable from t_i,

10. therefore tokens do not move through a route that does not lead to t_i.

11. therefore $\forall\ t_k \in |\ t_i\ \rangle,\ t_o \in |\ t_k\ \rangle$ or $\exists\ t_j\ |\ t_j = \bullet\bullet t_k$ and $t_o \notin |\ \bullet t_k\ \rangle$.

Hence, $\aleph_f$ is sound when these *necessary* conditions are satisfied.

Figure 4.2 shows a case when condition 4 of the Theorem is violated. Note that the number of activities allowed in the critical section is equal to $M(P_{co})$, which determines the **bandwidth** of the critical section. To extend the Theorem to allow N tokens to coexist in the critical section, Ψ is an N-bound $\mathrm{WF_{net}}$. To allow a single activity at a time, Ψ must be safe.

Lemma 4.1.2 If a $\mathrm{WFCS_{net}}$ $\aleph_f$ is sound then its $\mathrm{WF_{net}}$ $\aleph$ is sound and its control net Ψ is also sound.

We do not need to demonstrate this Lemma as the proof is implicit in the Theorem.

Lemma 4.1.3 For a sound $\mathrm{WFCS_{net}}$ $\aleph_f$, if the marking of $\aleph$ is $M(0)$ then the marking of Ψ is $M_c(0)$.

Proof 1. Since $\aleph_f$ is sound:

2. therefore t_i will eventually be fired and will consume a token from P_{co}, and,

3. $t_o \in |\ t_i\ \rangle$ [2].

4. Since t_o will eventually fire,

5. therefore a token ϵ will be produced in P_{ci}.

[2]Theorem 4.1.1

6. Since $\aleph$ is sound, therefore $o \;\in \mid t_o \rangle$.

7. Hence the marking of $\aleph$ will eventually be in i and o only,

8. Since Ψ is sound,

9. therefore the marking of Ψ will be in P_{co} only.

10. therefore if $M(\aleph) = M(0)$ then $M(\Psi) = M_c(0)$.

4.2 Quasi-Liveness and CS-property

The following Theorem binds between the soundness, Quasi-liveness and the CS-property of a WFCS_{net}:

Theorem 4.2.1 A WFCS_{net} is sound if and only if it is Quasi-Live and it satisfies the CS-property.

We start with proving that if a WFCS_{net}, $\aleph$ is sound, then it is *Quasli-live* and satisfies the *CS-property*.

Proof 1. Since $\aleph_f$ is sound,

2. therefore $\forall\ M_k \in \mid M_i,\ M_o\ \in\ \mid M_k \rangle$,

3. therefore $\forall\ t_k\ T,\ t_k$ is firable in a finit time,

4. therefore $\aleph_f$ is quasi-live.

5. Since $\aleph_f$ is quasi-live,

6. therefore $\forall \bullet t_k \in P, \bullet t_k$ is sufficiently marked,

7. therefore the CS-property is maintained.

Now we show the converse, namely, if the critical section workflow-net is quasi-live and maintains the CS-property then it is sound.

Proof 1. Since CS-property is satisfied,

2. therefore $\forall \bullet t \in T, \bullet t$ is always sufficiently marked.

3. Since $\aleph_f$ is quasi-live,

4. therefore $\forall\, t \in T,\ T$ is firable given marking M,

5. therefore $\bullet o$ is firable,

6. therefore $\forall\, \epsilon,\ \epsilon$ will evenyually be in o,

7. therefore $\aleph_f$ is sound.

4.3 Separability and Serializability

Separability and serializability are two dynamic behavior properties in workflow-nets as stated in definitions 3.4. Feedback affects such properties and therefore we investigate these features in this Section.

We define the N-Separability for a WFCS_{net} as follows:

Definition 7 The *degree of Separability* is the maximum number of activities that can be executed in a workflow while maintaining the property of separability.

Definition 8 N-Separability is a behavioral property which states that the behavior of a workflow-net with k tokens in the initial node can be seen as a combination of the behavior of N copies of the net, each of them with an average of $\frac{k}{N}$ tokens in the initial node, where N is the number of control tokens in P_{co}.

Note that N-Separability is fully maintained if $K \leq N$, where K is the number of activities in the system and N is the number of control tokens in P_{co}.

Theorem 4.3.1 *For a WFCS$_{net}$, $\aleph_f$, if $\aleph_f$ is sound, then it also satisfies the properties of Serialisability and N-Separability.*

Proof

1. Since $\aleph_f$ is sound,
2. therefore $\exists$ M_k such that $M_k \in \ \mid M(0) \cup M_c(0) \rangle$ and $M_o \in \ \mid M_k \rangle$,
3. therefore $\exists$ σ such that σ is a firing sequence, where $M(0) \xrightarrow{\sigma} M_o$.
4. Since $\aleph_f$ is Quasi-live[3], then $\exists$ $\{t_1, t_2, ..., t_n\} \in T$, such that $t_1 \xrightarrow{\sigma_1} t_2 \xrightarrow{\sigma_2} t_3 ... \xrightarrow{\sigma_{n-1}} t_n$, where n is a finite integer number and σ_{n-1} yields the system for marking M_o,
5. therefore $\sigma_1 \rightarrow \sigma_2 \rightarrow ... \rightarrow \sigma_{n-1} \equiv \sigma$,
6. therefore $\aleph$ is serializable.
7. Since it is serializable and from Lemma 4.1.3, It is an N-Separable.

Lemma 4.3.2 *A WFCS$_{net}$ $\aleph_f$ with m critical sections Ψ_i is k-separable, where k is the minimum initial marking of Ψ_i and $\Psi_i \in \Psi$ and $1 \leq i \leq m$.*

[3]See Theorem 4.2.1

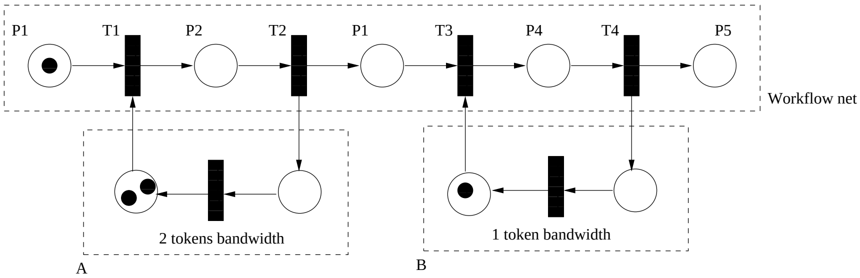


Figure 4.3: Bandwith of critical sections. Critical section A has a bandwidth of two tokens. Critical section B has a bandwidth of 1 token. The overall bandwidth of the workflow-net is then 1-token.

Lemma 4.3.2 shows that the bandwisth of the whole controlled workflow-net is equal to the minimum bandwidth of its critical sections as shown in figure 4.3.

4.4 Conclusion

In this Chapter a solution for introducing feedback and controlled loops into workflow-nets is presented. A Theorem of soundness for the proposed $\mathrm{WFCS_{net}}$ is given. We also proposed a Theorem for Quasi-Liveness and CS-property satisfaction. Serializability and separability were investigated.

Chapter 5

Cooperation Algebra

This Chapter describes our cooperative algebra for solving workflow problems. We propose operators that join a number of workflow-nets into a larger one. The algebra describes exactly how we build incident matrices that describe the Workflow-net compositions.

5.1 Cooperation Algebra

In this Section we show how a logical description of a plan is converted into our chosen representation of workflow-nets using the cooperative operator $\otimes$, as applied to create the incident matrices corresponding to logical operators. These are the *and* ($\wedge$), the *or* ($\vee$), the *then* ($\rightarrow$) ' and the *critical section* ($\lceil\rceil^n$) operators[1]. In this Chapter, we also study commutativity, associativity, and distributivity properties of the operators in two aspects, logical and structural. An operator can be logically associative but structurally non-associative. This applies to other

[1]The *not* operator or any higher level operator based on it (such as *xor*) are not used within this framework due to their lack of meaning in the workflow.

properties as will be seen later in this Chapter. An operator has a structural property if and only if applying this property on the operands does not affect the incident matrix built with the operator.

5.1.1 Predicates

In this framework, every predicate is transformed into a unit with a single input and a single output place. For instance, given predicate A, its incidence matrix is formed as

$$I_A = \begin{bmatrix} 1 \\ -1 \end{bmatrix}. \tag{5.1}$$

Initially, all predicates within the logical description of a cooperative plan are given incident matrices. Given an incident matrix I, we adopt the following definitions:

$$\begin{aligned} p(I) &\equiv \text{ number of rows in } I\text{, equivalent to the number of places,} \\ t(I) &\equiv \text{ number of columns in } I\text{, equivalent to the number of transitions,} \\ p_o(I) &\equiv \text{ row number of the output place in } I \textit{ and} \\ p_i(I) &\equiv \text{ row number of the input place in } I\text{,} \end{aligned}$$

which are used in the construction of incident matrices, resulting from applying the cooperation operator $\otimes$.

5.1.2 The $\wedge$ Operator

The *and* operator $\wedge$ joins the incident matrices of its predicates, yielding a new incident matrix describing the workflow-net resulting from applying the operator. In other words, $A \wedge B$ is equivalent to the following:

$$I_A \wedge I_B = \begin{bmatrix} 1 \\ -1 \end{bmatrix} \wedge \begin{bmatrix} 1 \\ -1 \end{bmatrix} = I_{A \wedge B}, \tag{5.2}$$

where $I_{A \wedge B}$ is the incident matrix. It is the $\wedge$ operator that gives rise to parallelism in the resulting workflow-net. For example, $A \wedge B$ signifies that A and B can be accomplished in parallel, given that enough resources with the required task coverage are available.

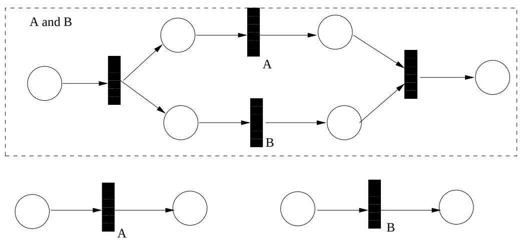


Figure 5.1: The application of AND operator to two workflow-nets.

With $(i = 0 \ldots p(I_A) + p(I_B) + 1, j = 0 \ldots t(I_A) + t(I_B) + 1)$, the incident matrix $I_{A \wedge B}$ is constructed as:

$$I_{A\wedge B} = \begin{cases} I_A(i,j) & \text{if } i < p(I_A) \text{ and } j < t(I_A) \\ I_B(i - p(I_A), j - t(I_A)) & \text{if } p(I_A) \le i < p(I_A) + p(I_B) \\ & \text{and } t(I_A) \le j < t(I_A) + t(I_B) \\ -1 & \text{if } i = p_i(I_A) \text{ and } j = t(I_A) + t(I_B) \\ 1 & \text{if } i = p_o(I_A) \text{ and } j = t(I_A) + t(I_B) + 1 \\ -1 & \text{if } i = p_i(I_B) \text{ and } j = t(I_A) + t(I_B) \\ 1 & \text{if } i = p_o(I_B) \text{ and } j = t(I_A) + t(I_B) + 1 \\ 1 & \text{if } i = p(I_A) + p(I_B) \text{ and } j = t(I_A) + t(I_B) \\ -1 & \text{if } i = p(I_A) + p(I_B) \text{ and } j = t(I_A) + t(I_B) + 1 \\ 0 & \text{otherwise} \end{cases} \tag{5.3}$$

For instance, with A and B as simple units, then

$$I_{A\wedge B} = \begin{bmatrix} 1 \\ -1 \end{bmatrix} \wedge \begin{bmatrix} 1 \\ -1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & -1 & 0 \\ -1 & 0 & 0 & 1 \\ 0 & 1 & -1 & 0 \\ 0 & -1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \end{bmatrix} \tag{5.4}$$

Lemma 5.1.1 *If $\exists\, A, B$ and A and B are two sound workflow-nets then $\exists\, C$ such that $C = A \wedge B$ and C is a sound workflow-net.*

We prove the previous Lemma as follows:

1. If the two workflow nets A and B are connected by a single input place p_i and a transition

t_i, then $\forall\ s \in S$ where S is the set of input tokens, s will end up being moved to both $i(A)$ and $i(B)$, where i is the input place of the workflow-net.

2. If A and B are sound workflow-nets, then $\forall\ s \in S$, s will end up being in $o(A)$ and $o(B)$, where o is the output place of the workflow-net.

3. $o(A)$ and $o(B)$ are the inputs to a transition t_o that is connected to the output place p_o, then $\forall\ s \in S$, s will end in p_o.

4. Therefore $A \wedge B$ is a sound workflow-net.

Operator $\wedge$ is logicallay and structuraly **commutative**, it is logically associative but structuraly non-associative, and it is logically distributive but structurlly non-distributive.

5.1.3 The $\vee$ Operator

The *or* operator $\vee$, not unlike the $\wedge$ operator, joins the incident matrices of two predicates to form a new incident matrix describing the resulting workflow net. This operator allows one part or another of the cooperative plan to be executed, depending on the results of prior execution. $A \vee B$ is equivalent to the following:

$$I_A \vee I_B = \begin{bmatrix} 1 \\ \\ -1 \end{bmatrix} \vee \begin{bmatrix} 1 \\ \\ -1 \end{bmatrix} = I_{A\vee B} \tag{5.5}$$

With $(i = 0 \ldots p(I_A) + p(I_B) + 1, j = 0 \ldots t(I_A) + t(I_B) + 3)$, the incident matrix $I_{A\vee B}$ is constructed as:

$$I_{A\vee B} = \begin{cases} I_A(i,j) & \text{if } i < p(I_A) \text{ and } j < t(I_A) \\ I_B(i-p(I_A), j-t(I_A)) & \text{if } p(I_A) \leq i < p(I_A)+p(I_B) \\ & \text{and } t(I_A) \leq j < t(I_A)+t(I_B) \\ -1 & \text{if } i = p_i(I_A) \text{ and } j = t(I_A)+t(I_B) \\ 1 & \text{if } i = p_o(I_A) \text{ and } j = t(I_A)+t(I_B)+2 \\ -1 & \text{if } i = p_i(I_B) \text{ and } j = t(I_A)+t(I_B)+1 \\ 1 & \text{if } i = p_o(I_B) \text{ and } j = t(I_A)+t(I_B)+3 \\ 1 & \text{if } i = p(I_A)+p(I_B) \\ & \text{and } j = t(I_A)+t(I_B) \text{ or } j = t(I_A)_t(I_B)+1 \\ -1 & \text{if } i = p(I_A)+p(I_B)+1 \text{ and } j \geq t(I_A)+t(I_B)+2 \\ 0 & \text{otherwise} \end{cases} \tag{5.6}$$

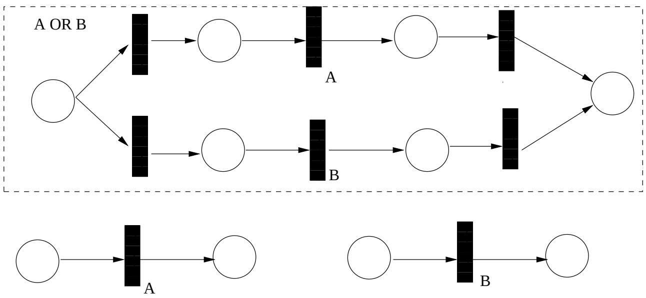


Figure 5.2: The application of OR operator to two workflow-nets.

For instance, with A and B as simple unit matrices, then

$$I_{A \vee B} = \begin{bmatrix} 1 \\ -1 \end{bmatrix} \vee \begin{bmatrix} 1 \\ -1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & -1 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & -1 & 0 & 0 \\ 0 & -1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & -1 \end{bmatrix} \tag{5.7}$$

Lemma 5.1.2 *If $\exists\, A, B$ and A and B are two sound workflow-nets then $\exists\, C$ such that $C = A \vee B$ and C is a sound workflow-net.*

We prove the above Lemma as follows:

1. Since the two workflow nets A and B are connected by a single input place p_i and two transitions $\bullet i(A)$ and $\bullet i(B)$ then, $\forall s \in S$ where S is the set of input tokens, s will be moved to either $i(A)$ or $i(B)$, where i is the input place of the workflow-net.

2. If A and B are sound workflow-nets, then $\forall s \in S$, s will end up being in $o(A)$ or $o(B)$, where o is the output place of the workflow-net.

3. If $o(A)$ and $o(B)$ are inputs to transitions $\bullet o(A)$ and $\bullet o(B)$ respectively that are both connected to the output place p_o, then $\forall s \in S$, s will end up being in p_o.

4. Therefore $A \vee B$ is a sound workflow-net.

Operator $\vee$ is logicallay and structurally **commutative**, it is logically associative but structurally non-associative, and it is logically distributive but structurally non-distributive.

5.1.4 The → Operator

The *then* operator → creates the sequential sections of cooperative plans between predicates. For instance, the plan $A \to B$ ensures that A is performed before B. The → operator joins the incident matrices of its predicates, creating a new incident matrix describing the workflow-net resulting from applying the operator. Hence $A \to B$ is equivalent to the following:

$$I_A \to I_B = \begin{bmatrix} 1 \\ -1 \end{bmatrix} \to \begin{bmatrix} 1 \\ -1 \end{bmatrix} = I_{A \to B} \tag{5.8}$$

With $(i = 0 \ldots p(I_A) + p(I_B) - 1, j = 0 \ldots t(I_A) + t(I_B))$, the incident matrix $I_{A \to B}$ is constructed as:

$$I_{A \to B} = \begin{cases} I_A(i,j) & \text{if } i < p(I_A) \text{ and } j < t(I_A) \\ I_B(i - p(I_A), j - t(I_A)) & \text{if } p(I_A) \leq i < p(I_A) + p(I_B) \\ & \text{and } t(I_A) \leq j < t(I_A) + t(I_B) \\ 1 & \text{if } i = p_o(I_A) \text{ and } j = t(I_A) + t(I_B) \\ -1 & \text{if } i = p_i(I_B) \text{ and } j = t(I_A) + t(I_B) \\ 0 & \text{otherwise} \end{cases} \tag{5.9}$$

For instance, with A and B as simple unit matrices, then

$$I_{A \to B} = \begin{bmatrix} 1 \\ -1 \end{bmatrix} \to \begin{bmatrix} 1 \\ -1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ -1 & 0 & 1 \\ 0 & 1 & -1 \\ 0 & -1 & 0 \end{bmatrix} \tag{5.10}$$

is the incident matrix $I_{A \to B}$.

Lemma 5.1.3 *If $\exists\, A, B$ and A and B are two sound workflow-nets then $\exists C$ such that $C = A \to B$ and C is a sound workflow-net.*

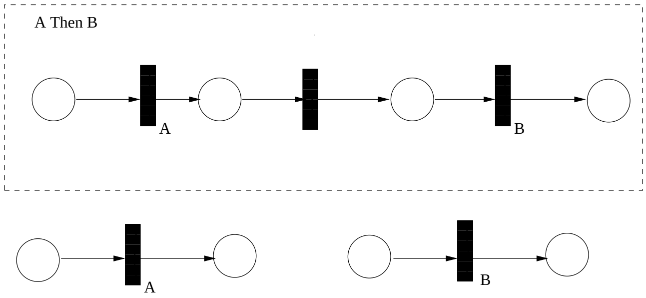


Figure 5.3: The application of THEN operator to two workflow-nets.

We prove the above Lemma as follows:

1. Since A and B are sound workflow-nets, then tokens will reach the output place of A,

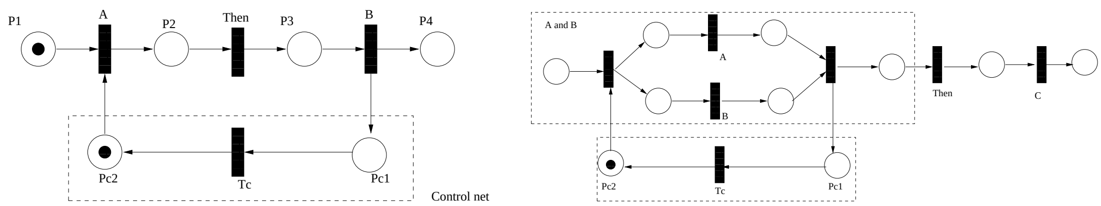
2. Since A and B are connected using a transition t with output place in A as an input of t and the input place of B as an output place of t, then $\forall s \in S$, where S is the set of input tokens, s will end up being in the output place of B.

3. Therefore $A \rightarrow B$ is a sound workflow-net.

Operator $\rightarrow$ is logicallay and structurally **non-commutative**, it is logically and structurally associative.

5.1.5 The $\lceil\rceil^n$ Operator

As seen before in Chapter 4, a critical section is a workflow-net or a sub-workflow-net that cannot serve more than n activities at a time, where n is an integer number, $1 \leq n \leq k$, and k is the maximum allowed number of concurrent activities in the system. It is worth noting that the critical section is defined by two or more transitions. Given the logic $A \rightarrow B$, applying a critical section on these two predicate, $\lceil A \rightarrow B \rceil^1$ is shown in Figure 5.4.



(a) **(b)**

Figure 5.4: Applying Critical sections on predicates: **a)**: $\lceil A \rightarrow B \rceil^1$ **b)**: $\lceil A \wedge B \rceil^1 \rightarrow C$.

The $\lceil\rceil^n$ operator limits the flow of tokens in the workflow-net by the number specified by n at any point in time. Having the logic $A \rightarrow B$ with an incident matrix of

$$I_{A \rightarrow B} = \begin{bmatrix} 1 & 0 & 0 \\ -1 & 0 & 1 \\ 0 & 1 & -1 \\ 0 & -1 & 0 \end{bmatrix} \tag{5.11}$$

, then $\lceil A \rightarrow B \rceil^n$ defines $A \rightarrow B$ as a critical section with maximum capacity of n tokens. The critical section binds the last transition and the first transition of the workflow-net. The incident matrix $\lceil A \rightarrow B \rceil^n$ is constructed as:

$$\lceil A \rightarrow B \rceil^n = \begin{cases} I_{A\rightarrow B} & \text{if } i < p(I_{A\rightarrow B}) \text{ and } j < t(I_{A\rightarrow B}) \\ 1 & \text{if } i = p(I_{A\rightarrow B}) \text{ and } j = t(I_{A\rightarrow B}) \\ -1 & \text{if } i = p(I_{A\rightarrow B}) + 1 \text{ and } j = t(I_{A\rightarrow B}) \\ -1 & \text{if } i = p(I_{A\rightarrow B}) \text{ and } j = t(I_{A\rightarrow B}) - 1 \\ 1 & \text{if } i = p(I_{A\rightarrow B}) + 1 \text{ and } j = 0 \\ 0 & \text{otherwise} \end{cases} \tag{5.12}$$

The incident matrix for $\lceil A \rightarrow B \rceil^n$ is:

$$I_{\lceil A\rightarrow B \rceil^n} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ -1 & 0 & 1 & 0 \\ 0 & 1 & -1 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & -1 & 1 \\ 1 & 0 & 0 & -1 \end{bmatrix} \tag{5.13}$$

The proof of soundness of this operator is found in Chapter 4.

5.2 A Cooperative Algebra Example

To illustrate the technique of computing incident matrices, consider the following cooperative plan:$A \wedge B \wedge ((C \vee D) \rightarrow E)$. The incident matrix of this plan is obtained by first assigning incident matrices to each of the predicates:

$$I_A = \begin{bmatrix} 1 \\ -1 \end{bmatrix}, \quad I_B = \begin{bmatrix} 1 \\ -1 \end{bmatrix}, \quad I_C = \begin{bmatrix} 1 \\ -1 \end{bmatrix}, \quad I_D = \begin{bmatrix} 1 \\ -1 \end{bmatrix} \text{ and } \quad I_E = \begin{bmatrix} 1 \\ -1 \end{bmatrix}.$$

Using standard precedence for logical operators, the suite of computations is as follows:

$$I_{A\wedge B} = \begin{bmatrix} 1 & 0 & -1 & 0 \\ -1 & 0 & 0 & 1 \\ 0 & 1 & -1 & 0 \\ 0 & -1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \end{bmatrix}, \quad I_{C\vee D} = \begin{bmatrix} 1 & 0 & -1 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & -1 & 0 & 0 \\ 0 & -1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & -1 \end{bmatrix},$$

$$I_{(C\vee D)\to E} = \begin{bmatrix} 1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & -1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 \end{bmatrix} \text{ and}$$

$$I_{A\wedge B\wedge((C\vee D)\to E)} = \begin{bmatrix}
1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
-1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & -1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 \\
0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & -1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & -1 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1
\end{bmatrix}.$$

The way these incident matrices are manipulated to produce a workflow-net corresponding to a cooperative plan (using the $\otimes$ operator) is elaborated in the description of the algorithm which follows in the next Chapter.

Chapter 6

Workflow-net Based Cooperation

In order to design a framework that is capable of supporting cooperation among a set of agents, the tasks to be performed by the system must be taken into consideration. The diversity of task types and constraints yield different designs. An example of this is agent polymorphism, which exists when two or more agents with different capabilities are able to complete the same task. A group of agents is said to be *homogeneous* if the capabilities of the individual agents are identical and *heterogeneous* otherwise. Heterogeneity introduces complexity because task allocation becomes more involved and agents need to model other individuals in the group.

6.1 Contribution

We aim to define a cooperative framework for robotic agents with the use of workflow-nets as defined by Aalst [56, 54]. Our main contribution consist of the application of workflow-nets to problems of cooperation among robotic agents. In this context, we define a cooperation operator (Section 6.5) which is used to compose agent capabilities expressed as workflow-nets into a cooperative, composed workflow-net. We demonstrate that this cooperation operator preserves the property of soundness, and that the framework is scalable to any number of agents with any number of capabilities. We also propose an algorithm which finds the minimal cost of the agent cooperation (Section 6.6).

We use workflow constructs to perform workflow-net compositions that are similar to those of Aalst [1, 55]. Our proposed cooperation operator results in performing common place and transition compositions.

We are interested in cooperating agents sharing their capabilities in the aim of achieving a cooperative plan. In that sense, unlike web service composition approaches [62, 33], we assume that compatibility is assured in the context of agents sharing their capabilities. In web service composition, the issue is one of composed and compatible service at design time while our approach for cooperating agents must determine the optimal fashion in which to share capabilities for the execution of a cooperative plan at a minimal cost.

Finally, Aalst states that for a composed workflow to be sound, every sub-workflow must end with a token in its output place [55]. Kindler argues that sub-workflows for which it is known that no token will reach their output places should not have undue influence on the soundness of the composed workflow [33]. In our framework, the composed, cooperative workflow is sound as per Aalst's criterion.

6.2 Preliminaries

We hypothesize that there is a set $\Lambda = \{\lambda_1, \lambda_2, \ldots, \lambda_m\}$ of primitive capabilities that cannot be fragmented into simpler capabilities and that the set of non-primitive action types $\epsilon \subset \Lambda$. Any action α from a robot at a given time is constructed from a list of primitive action types.

If a robot r_i from the set of cooperating robots $R = \{r_1, r_2, \ldots, r_n\}$ has plan d_j from the set of plans $D = \{d_1, d_2, \ldots, d_k\}$, then the robot can perform its plan on its own if and only if it meets the time constraints (if any), and the following equation holds:

$$\forall \alpha \in d_j : \alpha \in \omega(r_i), \tag{6.1}$$

where α is an action, and

$$\omega(r_i) = \{\mathrm{WF}_{\mathrm{net}1}, \mathrm{WF}_{\mathrm{net}2}, \ldots, \mathrm{WF}_{\mathrm{net}l}\} \tag{6.2}$$

is the action capability set of robot r_i, where $\mathrm{WF}_{\mathrm{net}i}$ are workflow-nets, and $d_j = \alpha_o(\cup\alpha_k)^*$, where $\alpha_o \neq \emptyset$ is a starting action and $(\alpha_k)^*$ is a set of following actions (which might be the empty set $\emptyset$).

Two robots r_i and r_k can cooperate to perform a desired plan d_j if they satisfy the task coverage property as follows:

$$\forall \alpha \in d_j : \alpha \in \omega(r_i) \cup \omega(r_k). \tag{6.3}$$

Robot r_k is a candidate for cooperation with robot r_i if and only if

$$\forall \alpha \in \Delta_j : \Delta_j = d_j - \omega(r_i)\, , \alpha \in \omega(r_k), \tag{6.4}$$

where Δ_j is the difference between the capabilities required to achieve plan d_j and the capabilities of robot r_i.

In our proposed framework, we use workflow-nets to model robots involved in cooperation and derive benefits from their structural and behavioral characteristics to build a protocol for cooperation. Conditions for an action to be taken are given by the input places to a transition and the results of performing the action are given by the output places from that same transition. Activities, which can be thought of as sets of actions performed by robots, are represented as tokens in workflow-nets.

6.3 Choice Dependency and Unit Similarity

We address the notions of choice-dependency and unit similarity as they are crucial concepts that affect the design of our framework. For instance, if two or more units among a set of workflow-nets are deemed similar, then they can be interchanged in order to accomplish the same task or part thereof. It is thus imperative to identify similar units in order to exploit parallelism and minimize the costs of cooperation.

Choice dependency occurs when two or more units share one or more input places. In such cases, soundness may not be ensured, as one or more of the choice-dependent units may not result in the presence of a token in the output place of the composed, cooperative workflow-net.

We define choice-dependency formally and, by design ensure that our framework avoids these problems by enforcing that the output place of the cooperative framework is reachable by all units. Additionally, we provide a technique to identify similar units in what follows[1].

Given a group of robots, their behavioral characteristics must be taken into consideration if the cooperation is to be successful. With that intent in mind, we divide a Petri-net into units v_i, where $1 \leq i \leq k$ and k is the number of units composing the Petri-net. A unit is a transition comprised of sets of input and output places which model an action, the conditions that must be satisfied prior to its execution, and the results of achieving the action, respectively. We proceed

[1] We inductively define units with the initial set of agent capabilities Λ as a basis.

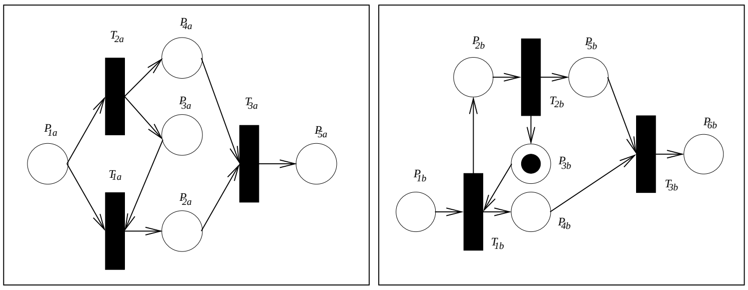


Figure 6.1: *The notion of choice dependency: Although $T_{1a} = T_{1b}$ and $\| \bullet T_{1a} \| = \| \bullet T_{1b}$, u_1 cannot replace u_3, since two transitions leave from P_{12}.*

with the mathematical definition of a unit:

Definition A unit is a tuple:

$$u_i = (\bullet T_i, T_i, T_i \bullet), \tag{6.5}$$

where T_i is a transition, $\bullet T_i$ the set of input places to T_i, and $T_i \bullet$ the set of output places to T_i.

The notion of choice dependence among units is relevant as it directly affects levels of cooperation. Units u_1 and u_2 are said to be choice-dependent if and only if their transitions share one or more input places. For instance, if unit u_1 and u_2 are choice-dependent, but unit u_3 is choice-independent, then unit u_1 cannot replace unit u_3 in its actions (and *vice-versa)*, as depicted in Figure 6.1.

Definition *A unit u is choice-independent if and only if the following condition holds:*

$$\bullet T_i \cap \bullet\ (T - T_i) = \emptyset, \tag{6.6}$$

where T is the set of transitions in a Petri-net, and $T_i \in T$.

If the unit is *choice-dependent*, then the set of *choice-dependency* is defined as:

$$\{T_j \mid T_j \in \{T - T_i\} \text{ and } \bullet T_i \cap \bullet T_j \neq \emptyset\} \tag{6.7}$$

and can be determined by satisfying the following condition:

$$W^+(P_j, T_i) - W^+(P_j, T_k) = 0 \tag{6.8}$$

$$\forall_{j=1}^{m} \; P_j \in \bullet T_i \cap \bullet T_k \;\text{ and }\; \forall_{k=1}^{m} \; T_k \in T,$$

where m is the number of places $P_j \in \bullet T_i$, k the number of transitions $T_k \in T$, and W^- the input incident matrix of the Petri-net.

Two units are identical if and only if they satisfy similarities in transition, pre-condition and post-condition. A transition similarity is defined by the action it belongs to. Two transitions T_1 and T_2 are similar if and only if $T_1 \in \lambda$ implies that $T_2 \in \lambda$, where λ is an action belonging to Λ. Pre-condition similarities are determined by satisfying

$$\Gamma(T_1) - \Gamma(T_2) = 0 \tag{6.9}$$

and post-conditions similarities, by satisfying

$$\Pi(T_1) - \Pi(T_2) = 0 \tag{6.10}$$

where $\Gamma(T_i)$ and $\Pi(T_i)$ are column vectors representing the input and output places to and from transition T_i, respectively.

Definition *A unit u_1 is similar to unit u_2 (denoted $u_1 \equiv u_2$) if and only if $\exists\, T_1 \in u_1$ and $\exists\, T_2 \in u_2 \mid \Lambda(T_1) = \Lambda(T_2)$ and $\bullet T_1 = \bullet T_2$.*

Definition *A unit u_1 is identical to unit u_2 (denoted $u_1 = u_2$) if and only if $\exists\, T_1 \in u_1$ and $\exists\, T_2 \in u_2 \mid \Lambda(T_1) = \Lambda(T_2)$ and $\bullet T_1 = \bullet T_2$ and $T_1\bullet = T_2\bullet$.*

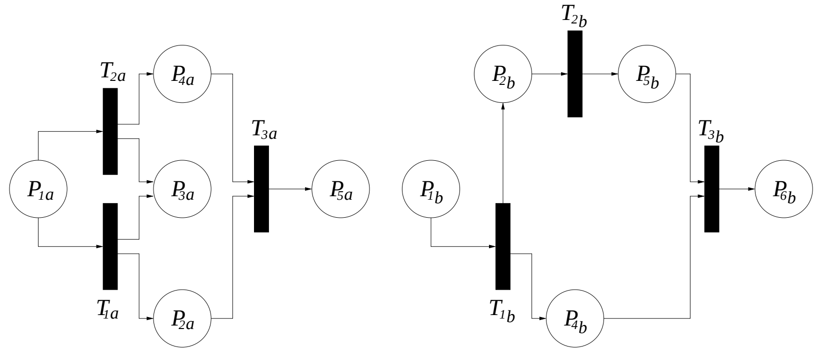


Figure 6.2: *Two different Petri-nets illustrating the concept of similarity: T_{1a} and T_{1b} perform the same task but are not similar because T_{1a} is choice-dependent while T_{1b} is not.*

An example of similar units from two different Petri-nets is given in Figure 6.2. While the concept of units is a step forward in defining cooperative processing, it is not practical, as most units in realistic situations are choice-dependent. Consequently, two or more choice-dependent units may find themselves exchanging actions (or tokens) more often than necessary. Hence, the success of cooperative choice-dependent units is not guaranteed. However, if there is a possibility to view the group of interdependent units as one composition, the process of cooperation becomes feasible and the success of the cooperative process is then guaranteed. Toward this end, we proceed with the definition of compositions of units within Perti nets:

Definition *A composition C is a set of joined units in a topology:*

$$C = \{U, P, F\}, \tag{6.11}$$

where U is a set of units, P a set of places, and $F \subseteq U \times P \cup P \times U$. Figure 6.3 illustrates similar compositions from two Petri-nets.

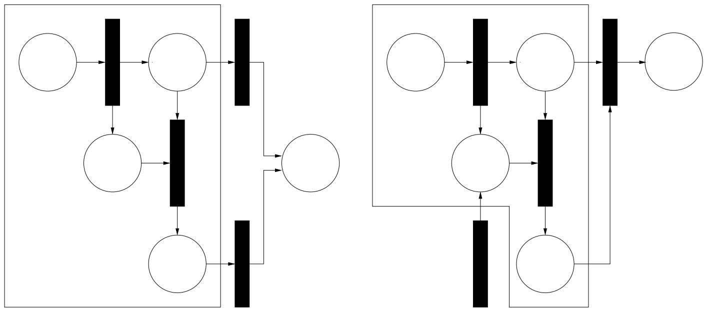


Figure 6.3: *Similar compositions in different Petri-nets. A composition is is a set of units in a Petri-net. In this example, while the two Petri-nets are different, there are inner compositions that are similar.*

Definition *A composition* $C_1 \subset C_2$ *if and only if* $\forall\ u_i \in C_1\ \ \exists\ u_j \in C_2 \mid u_i \equiv u_j$.

Definition *A composition* C_1 *is similar to* C_2 *if and only if* $\forall\ u_i \in C_1\ \ \exists\ u_j \in C_2 \mid u_i \equiv u_j$ *and* $\forall\ u_j \in C_2\ \ \exists\ u_i \in C_1 \mid u_j \equiv u_i$.

Definition *A composition* C_1 *is identical to* C_2 *if and only if* $\forall\ u_i \in C_1\ \ \exists\ u_j \in C_2 \mid u_i = u_j$ *and* $\forall\ u_j \in C_2\ \ \exists\ u_i \in C_1 \mid u_j = u_i$.

In other words, the preceding definitions extend the properties (similarity and equality) of units to compositions, among other things.

6.4 Redirecting Activities to Similar Units

One of the merits of this technique resides in its ability to use similar compositions in Petri-nets to perform one or more actions from the task under consideration in such a way as to allow its deadline to be met. Consider the example in Figure 6.4 and assume that there is a token

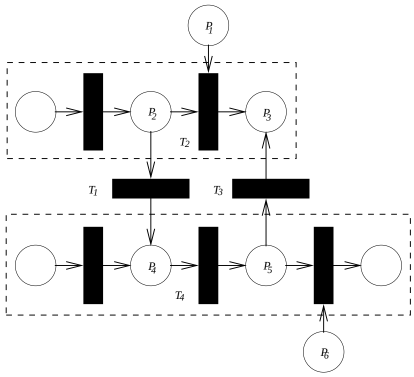


Figure 6.4: *Two cooperating Petri-nets. Transitions T_2 and T_4 are similar. Places P_1 and P_6 are routing places that must remain empty for the execution of the routing. In this example T_1 fires and moves the token to be executed by T_4 instead of T_2. After execution, the token is rerouted back to its parent workflow.*

that is going to miss its deadline in place P_2. If P_1 is empty, then transition T_1 is enabled and executed. As a result, the token under consideration is consumed from place P_2 and regenerated in P_4. When T_4 executes, then the token in P_4 is processed and appears in P_3. Following this sequence, T_3 executes, moves the token to P_3. The purpose of transitions T_1 and T_3 is to ensure that the migrating token goes to the desired route and returns, as opposed to being consumed by an undesired transition.

6.4.1 Examples of Cooperation

Suppose we have a workflow such as that of Figure 6.5 and two robots r_i and r_j. We assume that every transition in the workflow is in set $\omega(r_i)$. In the case where $\omega(r_j) = \omega(r_i) - \alpha$, where α is an action to reach a stack of objects, r_i is clearly unable to perform the plan on its own

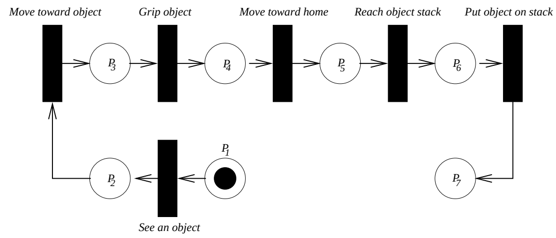


Figure 6.5: *A simple WF_{net} model for a robot activity which consists of 6 transitions, each representing a predefined action. Usually, models for robot activities are complex; this example shows how actions are mapped into transitions.*

and must ask for the cooperation of r_j to complete its mission, as depicted in Figure 6.6. If the function of both r_i and r_j is to grab objects from a loading zone and put them as stacks in another location, then r_j is able to achieve the required task on its own, whereas robot r_i is only able to get objects from the loading zone to a location near a stack. If the two robots cooperate, then whenever robot r_i grabs an object and transfers it to the home zone, it hands it to robot r_j if it is available, and r_j can put it on the top of a stack. If r_j is not available, then r_i waits until r_j becomes available. This cooperative protocol, as exemplified in Figure 6.6, shows that when robot r_i hands the object over to r_j, the token that represents the object is then given as the input node of the workflow representing r_j, so as to respect the requirement that any workflow has a single input entry point.

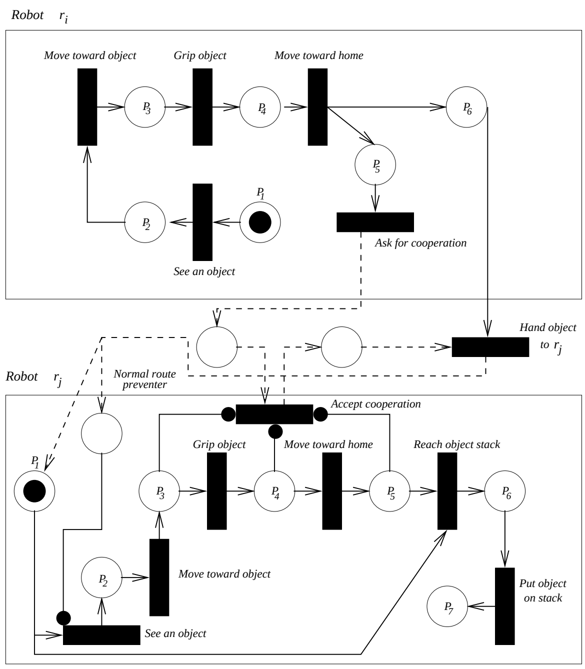


Figure 6.6: *A cooperative solution for robots r_i and r_j to collect objects and put them on a stack in a home zone. Robot r_i cannot complete its mission unless it cooperates with r_j. If robot r_i already has an object and is waiting in the home zone, then it hands the object directly to r_j through the execution of transition T_R. Following this, R_j puts the object on the stack. P_R is a routing place which when empty allows the input token to go directly to the Reach Object Stack transition. The arc ending with a black circle is an inhibitor. The dashed lines are the communication and interaction protocol between the robots*

6.4.2 Correctness of Framework

In order to show that the proposed framework is correct, we need to demonstrate that it yields the desired goals for cooperation. As previously stated, the provability problem from linear logic is a reachability problem in Petri-nets. Since we assume that agent capabilities can be expressed with workflow-nets ($\mathrm{WF}_{\mathrm{net}}$), then reachability for these workflow-nets is assured [54, 2]. However, we must guarantee that the proposed framework has the property of *soundness* [35, 50], as workflow-nets representing capabilities are assembled to form a cooperation plan, which is also a workflow-net. Hence, there is a need to demonstrate that the way by which the plan is constructed preserves soundness. The cooperative plan is the output of the cooperative framework.

Consider a cooperative framework among robots:

$$\Theta =< \Lambda, R, \Omega(R), D, S, \xi >, \tag{6.12}$$

where Λ is the set of primitive action types, R the set of cooperating robots,

$$\Omega(R) = \{\omega(r_1), \omega(r_2), \ldots, \omega(r_n)\} \tag{6.13}$$

the set of all robot capabilities, and D the set of plans to be performed by the set of robots. The set of all similarities between robot capabilities is defined as:

$$S = \{S_1, S_2, \ldots, S_{n(n-1)}\}, \tag{6.14}$$

where

$$S_k = \mathrm{WF}_{\mathrm{net}_i} \cap \mathrm{WF}_{\mathrm{net}_j}, \tag{6.15}$$

$\forall\ \mathrm{WF}_{\mathrm{net}_i} \in \omega(r_i)$ and $\forall\ \mathrm{WF}_{\mathrm{net}_j} \in \omega(r_j)$. $\xi = \{\xi_1, \xi_2, \ldots, \xi_z\}$ is the set of workflows that bind two or more different workflows from two or more robots.

A framework is sound if any valid input plan can be carried out successfully, under the hypothesis that the set of robot capabilities satisfies the task coverage requirement. For mathematical convenience, we add a single input place p_i and a single output place p_o.

Theorem 6.4.1 *A cooperation framework Θ is sound if and only if the following conditions are satisfied:*

1. $\forall\ WF_{net} \in \Omega(R),\ WF_{net}$ *is sound,*

2. $\forall\ \lambda \in \Lambda, \lambda \in \Omega(R),$

3. $\forall\ d \in D,\ \exists\ WF_{net_i} \otimes WF_{net_j} \mid d$ *is executable and*

4. $\forall\ \xi_k \in \xi,\ \xi_k$ *is a sound workflow-net*

where WF_{net_i} and WF_{net_j} are workflow-nets representing capabilities from agents i and j, and $\otimes$ is the cooperation operator, as described in Section 6.5.

6.4.3 Demonstration

1. Since Θ is sound, then $o\ \in\ \mid i\,\rangle$ and $\forall\ d\ \in\ D$, d will eventually reach o, regardless of the $\mathrm{WF_{net}}$ it goes through. Hence, we have $\forall\ \mathrm{WF_{net}}\ \in\ \Omega(R)$, $\mathrm{WF_{net}}$ is a sound workflow-net.

2. Since $o\ \in \mid i\,\rangle$, $d = \{\alpha_o \cup (\alpha_i^*)\}$, and that d will eventually execute (resulting as a token in o), we thus have $\forall\ \lambda \in \alpha, \lambda \in \omega(R)$, and consequently $\forall\ d \in D,\ \exists\ \mathrm{WF}_{\mathrm{net}_i} \otimes \mathrm{WF}_{\mathrm{net}_j} \mid d$ is executable.

3. The definition of soundness $\forall\ M\ \in\ \mid i\,\rangle$, $o\ \in\ \mid M\,\rangle$ implies $\forall\ \xi_i\ \in\ \xi,\ \ \xi_i$ is a sound workflow-net.

Therefore, for sound Θ, the four necessary conditions are satisfied. We now show the converse, namely that if the four conditions are satisfied, the framework is sound:

1. Since $\forall$ $\mathrm{WF_{net}} \in \Theta(R)$, $\mathrm{WF_{net}}$ is a sound workflow-net, we have $\forall$ $\xi_i \in \xi$, ξ_i is a sound workflow-net.

2. We have $o \in |\ i\ \rangle$ and, since $\forall$ $\lambda \in \alpha, \lambda \in \Theta(R)$, we obtain $\forall$ $d \in D$, $\exists$ $\mathrm{WF_{net_i}} \otimes \mathrm{WF_{net_j}} \mid d$ is executable and we conclude that Θ is sound.

6.5 The Cooperative Operator

The cooperative operator embodies the cooperation framework, as it joins (or composes) two cooperative frameworks into one. This joint framework must be sound for it to represent a valid cooperation framework. In light of this, the cooperative operator $\otimes$ must satisfy a number of conditions. For instance, if $\mathrm{WF_{net_k}} = \mathrm{WF_{net_i}} \otimes \mathrm{WF_{net_j}}$, then the following properties must apply:

1. $\mathrm{WF_{net_i}}$ and $\mathrm{WF_{net_j}}$ are sound

2. $\mathrm{WF_{net_k}}$ is a workflow-net with two special places i_k and o_k

3. $\forall$ $\zeta \in \mathrm{WF_{net_i}} \otimes \mathrm{WF_{net_j}}$, ζ is sound

4. $\bullet i_k = \emptyset$, $o_k \bullet = \emptyset$

6.6 Description of Algorithm

In this Section we describe the complete algorithm used to derive a cooperative workflow net from a plan expressed with the notation used in Section 5.1.

It is of note to consider that this algorithm generates a cooperative, composed workflow which contains all the possible cooperative scenarios. Consequently, the algorithm implements a back-tracking scheme allowing it to determine the minimal cost cooperative path from the workflow-net.

We begin by first defining the concepts and data structures used in the algorithm.

6.6.1 Definitions

1. Stages constitute the representation we use to express the order of computation of a plan, based on standard operator precedence.

2. Plan stages $G = \{g_1, g_2, \ldots, g_n\}$, where n is the number of stages and stage i, denoted as g_i, is a set of predicates which belongs to plan P.

3. A stage assignment A is a two-dimensional array with its number of rows equal to the number of stages n. Each row has a number of tuples (columns) equal to the assignment of this stage. Example: $A(0)$ is the assignment of the first stage and is of the form $(r_1, P_r(2)), (r_2, P_r(1)), \ldots, (r_m, P_r(k))$, where m is the number of agent assignments in this stage and k is the last assigned predicate. $A(0, 1)$ is the second assignment of the first stage which is $(r_2, P_r(1))$ in this example. This tuple signifies that agent r_2 is dedicated to execute predicate $P_r(1)$.

4. A two-dimensional matrix M with number of rows equal to number of robots involved in the current stage, and number of columns equal to the number of predicates in the plan. Essentially, $M(i, j)$ is the cost to execute predicate $P_r(j)$ with agent r_i.

5. $M(i)$ (the i^{th} row of M) is a vector representing the costs of the capabilities of agent r_i to execute the plan predicates.

6. $M^T(i)$ (the i^{th} column of M) is a vector representing the costs of predicates when executed by different agents.

7. A predicate $P_r(i)$ is affected by the *then* operator if and only if there is a rule in the plan that is in the form of $(P_r(k)(\{\wedge| \vee | \rightarrow\}P_r(j))^* \rightarrow P_r(i)$. We denote an operator $P_r(i)$ affected by a *then* operator using the symbol $P_r(i)^{\rightarrow}$, and $P_r(i)^{\nrightarrow}$ if it is not. A predicate that is affected by the $\rightarrow$ operator has its execution delayed by predicates prior to the operator in the plan. If $P_r(i) \rightarrow P_r(j)$ then $P_r(i)$ is denoted as $P_r(i)^{\leftarrow}$.

8. $|S_t(i)|$ is the cardinality of $S_t(i)$.

9. AgentIndex is a temporary agent index value in A.

10. (AgentIndex, $*$) $\in A(i)$ represents all tuples in stage $i \in A$ that has an agent with an index of AgentIndex.

6.6.2 Complete Algorithm

INPUT: A plan $P = \{P_i((\wedge|\vee| \rightarrow)P_j)^*\}$ where P_i and P_j are predicates, and a group of agents $R = \{r_1, r_2, \ldots, r_n\}$, each with a set of capabilities described as in (6.2)

OUTPUT: A cooperative framework Θ as described in (6.12)

---

for all $r_i, r_j \in R$ **do**

Calculate similarity $S(r_i, r_j)$.

end for

Test task coverage according to (6.3), end if not satisfied.

$i = 1$

for all predicate $P_r(j)$ and $P_r(i)^{\nrightarrow}$ **do**

$g_i = g_i \cup Pr(j)$

end for

Label: 1

Build matrix M for stage g_i

for $k = 1$ to $|g_i|$ **do**

Label: 2

AgentIndex = index of $\min\{M^T(k)\}$, where $\min\{M^T(k)\} \neq \infty$

if (AgentIndex, $*$) $\in A(i)$ **then**

M^T(AgentIndex, index of $\min\{M^T(k)\}) = \infty$

Goto Label 2

else if AgentIndex $= \emptyset$ **then**

Backtrack to $k - 1$

Choose the next least cost

else

$A(i) = A(i) \cup (\text{AgentIndex, index of } \min\{M^T(k)\})$

end if

end for

$i = i + 1$

if $\exists \; P_r(j), P_r(k) \;|\; P_r(j)^{\rightarrow}$ and $P_r(k)^{\rightarrow}$ and $P_r(k) \in g_{i-1}$ **then**

$g_i = g_i \cup P_r(j)$

Goto Label 1

end if

if $\exists \; r_u \in R \;|\; (u, *) \notin A$ and $\exists \; r_k \;|\; (k, C) \in A$ and $S(r_u, r_k) = C$ **then**

$A(\text{index}(k)) = A(\text{index}(k) \cup (r_u, C)$, where C is the set of capabilities of r_k in stage index(k)

and S is the set of similarities found with Definitions (6.3) and (6.3).

end if

for all $P_{rj} \in g_i$ **do**

Create a unit U_j as per Definition (6.5)

end for

for all $U_p, U_q \mid (P_{rp} \; \{\wedge| \vee | \rightarrow\} P_{rq}) \; \cap P \neq \emptyset$ **do**

Compute $\Theta = \Theta \otimes (U_p \otimes U_q)$ as defined in Section 5.1

end for

Output Θ

6.7 Framework Scalability

Scalability refers to the efficiency with which the system operates when the number of agents increases. Scalability is an important issue and constitutes a measure of the quality of the design of the multi-agent system. Our approach guarantees scalability as we demonstrate using mathematical induction (See Section 6.8 for a formal demonstration).

To prove our claim, we use the operator $\otimes$ presented earlier, which represents the cooperation framework. The operator $\otimes$ joins two cooperative frameworks, yielding a third framework as a result. The joined platforms must be sound, as shown in the proposed Theorem, to yield a valid, sound cooperative framework. The operator is initially applied on two robots as a base case, since a single robot may be thought of as a cooperative framework onto itself:

$$\Theta =< \Lambda, R, \Omega(R), D, \emptyset, \emptyset >, \tag{6.16}$$

where S and ξ are both $\emptyset$. Since the cooperative framework contains a single robot, the similarity set S is $\emptyset$ and the set of workflows that bind two or more different workflows from two or more robots ξ is also $\emptyset$.

While the framework is shown to be scalable, it is not guaranteed to yield the best performance in the case of n heterogeneous robots, where $n > 2$, since the selection of a cooperation robot pair among a set of candidate robots highly affects future plans. However, an optimal solution is obtained in the precise case when plans remain static during their execution since a linear programming technique is applied with respect to the costs of the transitions in the workflow-net.

6.8 Proof of Scalability

We prove the scalability of the platform using mathematical induction as follows:

1. Base case: Given two robots r_1, r_2 and a plan D such that $\forall\ \lambda \in D, \lambda \in \omega(r_1) \cup \omega(r_2)$ and $\forall\ \mathrm{WF}^i_{\mathrm{net}} \in \bigcup_{i=1}^{2} \omega(r_i), \mathrm{WF}^i_{\mathrm{net}}$ is a sound $\mathrm{WF}_{\mathrm{net}}$, $\exists\ \Theta$ such that $\Theta = r_1 \otimes r_2$ and Θ is a sound cooperation framework. The proof of this base step is the same proof as above.

2. Inductive hypothesis: Given k robots and a plan D such that $\forall\ \lambda \in D, \lambda \in \bigcup_{i=1}^{k} \omega(r_i)$ and $\forall\ \mathrm{WF}^i_{\mathrm{net}} \in \bigcup_{i=1}^{k} \omega(r_i), \mathrm{WF}^i_{\mathrm{net}}$ is a sound $\mathrm{WF}_{\mathrm{net}}$, $\exists\ \Theta_k$ such that $\Theta_k = ((...((r_1 \otimes r_2) \otimes r_3)... \otimes r_{k-1}) \otimes r_k)$ and Θ_k is a sound cooperation platform.

3. Inductive step: Given $k + 1$ robots and a plan D such that $\forall\ \lambda \in D, \lambda \in \bigcup_{i=1}^{k+1} \omega(r_i)$ and $\forall\ \mathrm{WF}^i_{\mathrm{net}} \in \bigcup_{i=1}^{k+1} \omega(r_i), \mathrm{WF}^i_{\mathrm{net}}$ is a sound $\mathrm{WF}_{\mathrm{net}}$. It is required to prove that $\exists\ \Theta_{k+1}$ such that $\Theta_{k+1} = (((...((r_1 \otimes r_2) \otimes r_3) \ldots \otimes r_{k-1}) \otimes r_k) \otimes r_{k+1})$ and Θ_{k+1} is a sound cooperation framework.

Since $\forall\ \lambda \in D$, $D \in \bigcup_{i=1}^{k+1} \omega(r_i)$, then $\lambda \in \omega(\Theta_k \otimes r_{i+1})$. Therefore

$$\lambda \in \omega(\Theta_{k+1}). \tag{6.17}$$

Since $\forall\ \mathrm{WF}^k_{\mathrm{net}} \in \Theta_k, \mathrm{WF}^k_{\mathrm{net}}$ is a sound $\mathrm{WF}_{\mathrm{net}}$ and $\mathrm{WF}^{k+1}_{\mathrm{net}} \in r_{k+1}$ is also a sound $\mathrm{WF}_{\mathrm{net}}$, and from the definition of operator $\otimes$, then $\forall\ \zeta \in \Theta_{k+1}, \zeta$ is sound. The cooperative framework Θ_{k+1} satisfies the conditions of the Theorem and hence it is sound.

From the last proof we obtain the following Lemma:

Lemma 6.8.1 *Given n robots and an executable plan D such that $\forall\ \epsilon \in D, \epsilon \in \bigcup_{i=1}^{n} \omega(r_i)$ and $\forall\ WF^i_{net} \in \bigcup_{i=1}^{n} \omega(r_i), WF^i_{net}$ is a sound WF_{net}, $\exists\ \Theta_n$ such that $\Theta_n = r_1 \otimes r_2 \otimes r_3 ... \otimes r_{n-1} \otimes r_n$ and Θ_n is a sound cooperation platform.*

6.9 Experimental Simulations

We built a simulator for the cooperation framework, in which the algorithm described in Section 6.6 is implemented. Our goal is to empirically demonstrate that our definition of cooperation is correct and that plans can be adequately established.

6.9.1 Experimental Set-Up

The input to the simulator consists of a plan in the form of a linear logic expression with operators as described in Section 5.1. Other input parameters consist of a set of agents, each with a set of capabilities, expressed as workflow-nets. Each capability corresponds to one action defined in the plan, along with the cost associated with performing that action. Actions that are not part of an agent's set of capabilities have their cost set to infinity. The simulator assigns costs in the following manner: first a uniformly distributed random variable is used to determine the initial set of capabilities for each agent. When an agent is assigned a capability, the cost for its execution is randomly determined with a normally distributed variable.

Once the agent capabilities are set, the simulation evaluates the task coverage. If the generated agent capabilities are insufficient to provide a complete task coverage, the simulator terminates. Otherwise, the cooperative plan is constructed (using the algorithm in Section 6.6) and executed. The execution time is computed simply as the total execution cost in the cooperative workflow-net.

6.10 Limitations

Our proposed framework for cooperation has a number of limitations, which we proceed to describe.

1. The input plan must be expressed with linear logic. Cooperative situations that cannot be expressed as such, cannot be dealt with.

2. The framework implicitly assumes that agent capabilities are sufficient to provide task coverage. In cases where agent capabilities do not provide an adequate task coverage, the cooperative plan will not terminate.

3. The framework poses the hypothesis that every agent action will be successful. In practice, when this is not the case, no mechanism is provided to remedy the situation and the cooperative plan may fail.

4. Any change in agent status cannot be considered while the cooperative plan is being determined. For instance, such changes may be modifications to agent capabilities. When capabilities change, the algorithm (from Section 6.6) must be executed again.

We present the results of 2 experiments to demonstrate our algorithm. The 1^{st} experiment was designed to demonstrate the way by which the framework exploits parallelism in plan execution. The plan to execute is expressed as

$$((((A \vee E) \rightarrow (B \vee C)) \rightarrow D) \wedge (G \rightarrow F)) \tag{6.18}$$

where each predicate represents a unique robot capability. For this plan, 7 different capabilities corresponding to the predicates A, B, C, D, E, F, and G are required for its execution. We

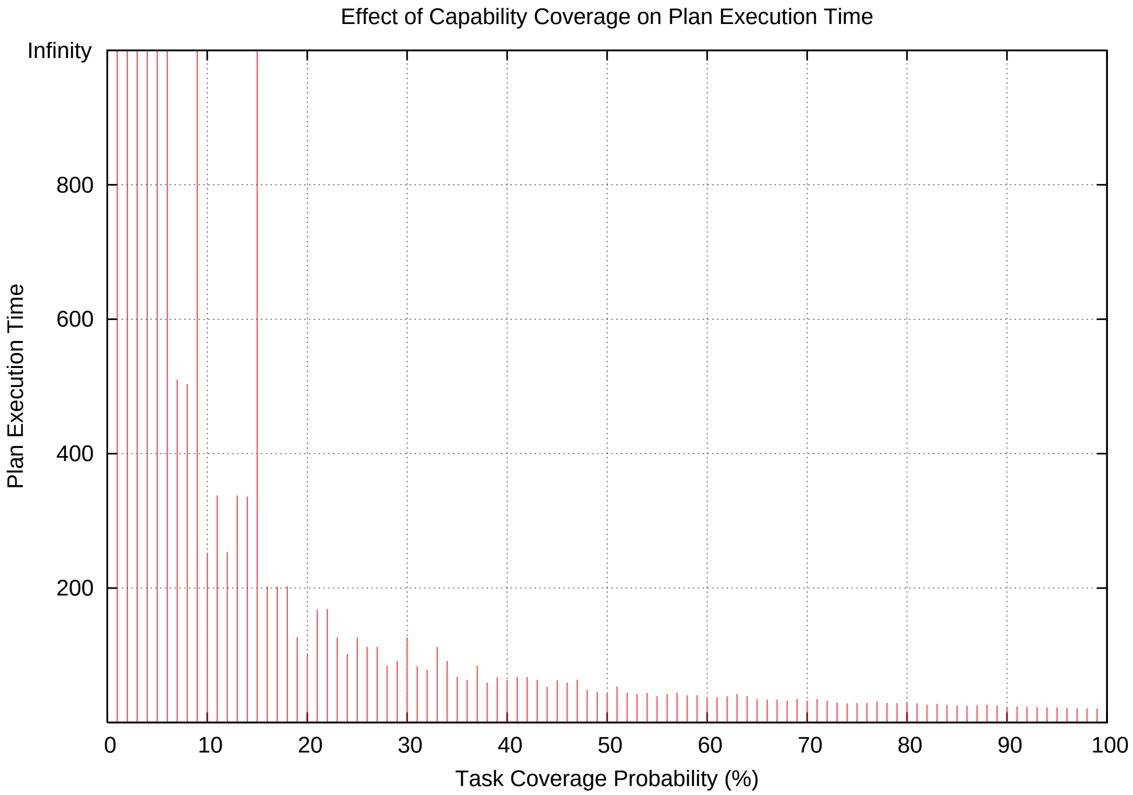


Figure 6.7: *Plan execution times versus probabilistic agent task coverage for a group of 50 robots.*

used 50 agents in 100 simulations, where we controlled the probability of an agent to possess each capability. For instance, for a task coverage probability of 50%, each one of the 50 agents would have a 50% chance of possessing each of the 7 capabilities. This experiment was performed for task coverage probabilities from 1 to 100%. As expected, with a 100% task coverage probability, each agent possesses all the 7 capabilities, resulting in full parallelism of execution, while respecting the flow constraints of the plan. The time units are expressed in terms of transition costs in the workflow nets. Note that these could express other measures. Figure 6.8 shows plan execution times for this set of simulations. As expected, times for low task coverage probabilities are high and sometimes infinite in cases when the probabilistic attribution of agent capabilities is insufficient to complete the plan. It is also observed that, as the probabilistic task coverage increases for each agent, the execution time decreases in what seems to be a negative exponential function. This is in part due to the logical structure of the plan, expressed with (6.18), which allows for parallelism. Conversely, execution times for a plan such as

$$A \rightarrow B \rightarrow C \rightarrow D \rightarrow E \tag{6.19}$$

would turn out as constant, or infinite, when the sum of probabilistic task coverages of agents is insufficient.

Our 2^{nd} of experiment explores the effects of varying the numbers of agents on plan execution times. As expected, execution times are shortened by increasing numbers of agents. Figure 6.9 a) depicts this situation, where a growing number of agents significantly reduces execution times. Figure 6.9 b), showing the results where the probabilistic task coverages are insufficient to complete the plan, and Figure 6.9 c), showing the converse, demonstrate that in the particular case of this plan, extended agent capabilities reduce execution times more drastically than the number of robots. In this particular case, the framework favors agents with better task coverage than number of robots for plan execution time reduction.

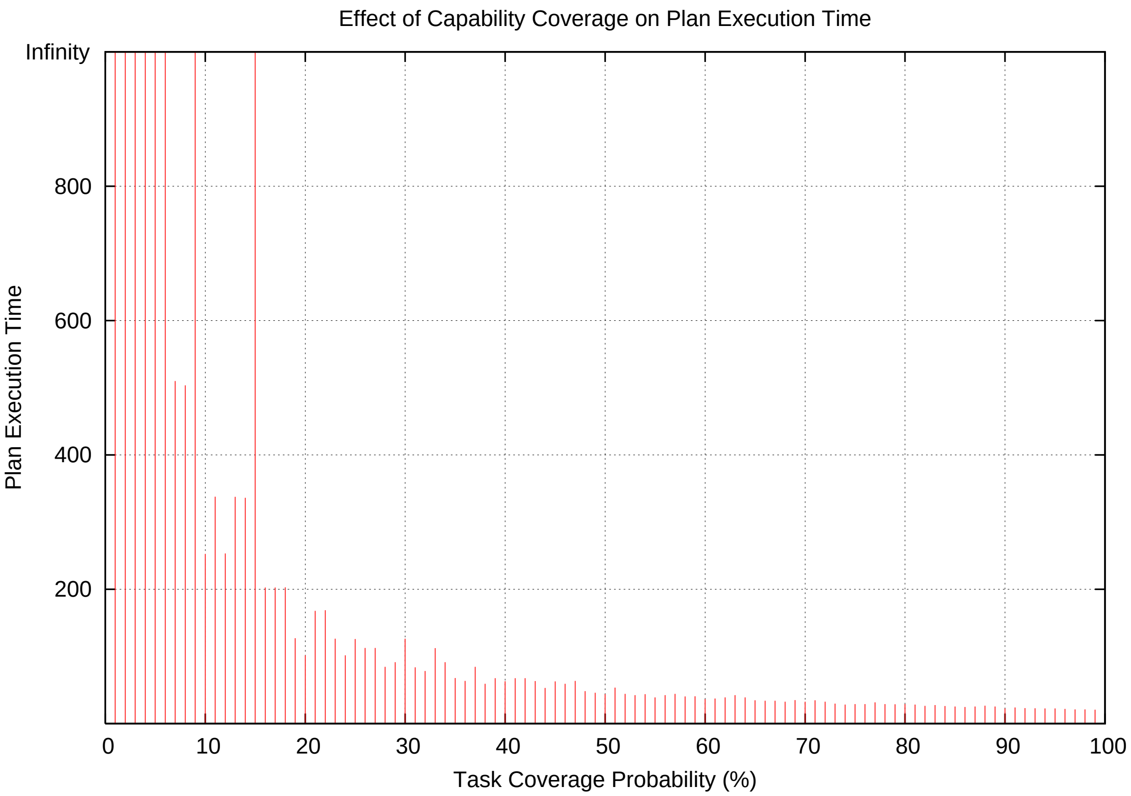


Figure 6.8: *Plan execution times versus probabilistic agent task coverage for a group of 50 robots.*

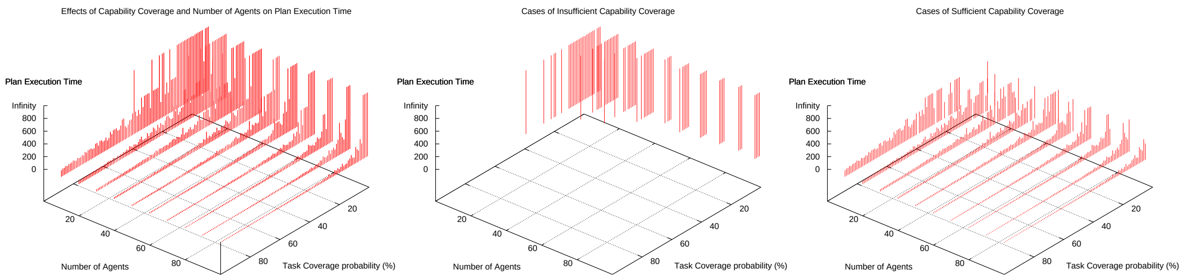


Figure 6.9: **a) (left):** *Plan execution times versus number of agents and probabilistic task coverage.* **b) (center):** *Insufficient task coverage cases.* **c) (right):** *Sufficient task coverage cases.*

Chapter 7

Conclusion

Workflow-net (which is an extention of Petri-nets) is one of the modelling techniques that describes distributed and parallel systems. Petri-nets have very powerful mathemnatical properties that helps not only the modelling of the system, but also the analysis of its static and run time behavior. Petri-nets in general and workflow-nets in particular are very good candidates to solve such problems. Workflow-nets (when properly designed) guarantee the soundness of the systems it models. In this Thesis, several frameworks are proposed for task syncronization and cooperation among a set of agents. In Chapter 4, the concept of the workflow critical section is presented. A critical section is a set of connected transitions that have an upper limit on the number of activities they can serve at any given time. A framework has been proposed to satisfy such a limitation and give feedback, whenever an activity passes through this critical section. In general, feedback in workflow-nets has not been advised as it threatens the soundness of the workflow-net. The proposed framework satisfies the soundness constraint if the workflow-net design follows the conditions proposed in Theorem 4.1.1. The soundness of the $\mathrm{WFCS}_{\mathrm{net}}$ also depends on the Quasi-Livness and the CS-property of the workflow-net. Quasi-Liveness means that for any transition t in the workflow-net, t is firable in a finite time. Control Siphon property (CS-property) states that for every siphon s in the workflow-net, s remains sufficiently marked to guarantee the reachability of all transitions in the workflow-net. Theorem 4.2.1 states that Quasi-Liveness and CS-property are necessary conditions for the soundness of the WFCS_{net}. Separability and Serializability are also studied for WFCS_{net}. Separability is the capability of tracing the execution of n activities in a workflow-net as the trace of each activity within a number of WFCS_{net}. This property ensures that the behavior of the workflow-net due to the current activity does not impact the behavior of the same workflow-net for the next activity. Serializability is the capability to see the trace of several activities being processed at the same time, as if would be a set of traces of activities that have been processed in a serial manner. Theorem 4.3.1 shows the relationship between the soundness of the WFCS_{net} and its serializability and separability.

Chapter 5 presented the algebraic operators that allow a given plan to be converted into a workflow-net. The logic has operators (and, or, then). Transformation functions are proposed that convert logical expressions into workflow-nets.

Chapter 6 demonstrated the proposed cooperative framework. The definition of cooperation was stated and the foundation on how to choose cooperative partners was proposed. The main foundation we wished to satisfy was task coverage. Task coverage is a concept that determines whether the task is achievable or not using the set of available agents. Workflow-net constructs are proposed in terms of units and composites. A unit is an atomic unit of a workflow-net that can be involved in cooperative behaviour. A composite is a non-empty set of units connected together. Formal methods were proposed to determine similar and identical units and composites.

We also showed how workflow-nets could be connected together to build a cooperative plan to execute certain common tasks among agents. Theorem 6.4.1 proposed a cooperative framework that guaranteed the soundness of the cooperative plan execution. Simulation experiements showed that the times for low task coverage probabilities are high and sometimes infinite in cases where the probabilistic agent capabilities is insufficient to complete the plan. It was observed that, as the probabilistic task coverage increased for each agent, the execution time decreased, as expected. This is in part due to the logical structure of the plan, which allows for parallelism. Our second set of experiments explores the effects of varying numbers of agents on plan execution times. As expected, execution times were shortened by increasing the number of agents.

Bibliography

[1] W.M. Van Der Aalst. The application of petri nets to work-flow management. *Journal of Circuits Systems and Computers*, 8(1):21–66, 1998.

[2] W.M. Van Der Aalst, M. Weske, and G. Wirtz. Advanced topics in work-flow management: Issues, requirements and solutions. *Journal of Integrated Design and Process Science*, 7(3):49–77, 2003.

[3] A. Al-Jumaily and S. Kozak. Behaviour based multi robot cooperation by target/task negotiation. In *proceedings of the 2004 IEEE conference on Robotics, Automation and Mechatronics*, pages 661–666, December 2004.

[4] R. Alami, S. Fleury, M. Herbb, F. Ingrand, and F. Robert. Multi-robot cooperation in the martha project. *IEEE Robotics and Automation Magazine*, 5(1):36– 47, March 1998.

[5] R. Alami, F. Ingrand, and S. Qutub. Planning coordination and execution in multi-robots environment. In *proceedings of the 1997 IEEE International Conference on Robotics and Automation*, pages 525– 530, July 1997.

[6] R. Alami, F. Robert, F. Ingrand, and S. Suzuki. Multi-robot cooperation through incremental plan-merging. In *Proceedings of the 1995 IEEE International Conference on Robotics and Automation*, pages 2573–2579, May 1995.

[7] L. Aldred, W. van der Aalst, M. Dumas, and A. ter Hofstede. Communication abstractions for distributed business processes. In *19th Int. Conf. on Advanced Information Systems Engineering*, pages 409–423, 2007.

[8] T. Arai, E. Pagello, and L.E. Parker. Editorial: Advances in multi-robot systems. *IEEE Trans. On Robotics and Automation*, 18(5):655–661, October 2002.

[9] T. Balch and L.E. Parker. *Robot Teams: From Diversity to Polymorphism*. A.K. Peters, Ltd., 2002.

[10] P. Baldan, A. Corradini, H. Ehrig, and R. Heckel. Compositional semantics for open petri nets based on deterministic processes. *Journal of Mathematical Structures in Computer Science*, 15(1):1–35, February 2005.

[11] P. Baldan, A. Corradini, H. Ehrig, and B. Konig. Open petri nets: Non-deterministic processes and compositionality. *Lecture Notes in Computer Science, Graph Transformations*, 5214:257–273, 2008.

[12] K. Barkaoui, P. Dechambre, and R. Hachicha. Verification and optimization of an operating room workflow. In *Proceedings of the 35th Annual International Conference on Systems Science*, pages 2581–2590, 2002.

[13] John L. Barron. Dialogue Organization and Structure for Interactive Information Systems. Master's thesis, Computer Science Department, University Of Toronto, Toronto Ontario, 1980.

[14] A. Borkowski, M. Gnatowski, and J. Malec. Mobile robot cooperation in simple environments. In *Proceedings of the Second International Workshop on Robot Motion and Control*, pages 109–114, October 2001.

[15] S.C. Botelho and R. Alami. M+: A scheme for multi-robot cooperation through negotiated task allocation and achievement. *Robotics and Automation, 1999. Proceedings. 1999 IEEE International Conference on*, 2:1234–1239, 1999.

[16] Y.U. Cao, A.S. Fukunaga, and A.B. Kahng. Cooperative mobile robotics: Antecedents and directions. *Autonomous Robots*, 4(5):7–27, 1997.

[17] L. Chaimowicz, M.F.M. Campos, and V. Kumar. Hybrid systems modeling of cooperative robots. In *Proc. IEEE Int. Conf. on Robotics and Automation*, pages 4086–4091, Taipei, Taiwan, September 2003.

[18] L. Chaimowicz, T. Sugar, V. Kumar, and M.F. Campos. An architecture for tightly coupled multi-robot cooperation. In *proceedings of the 2001 IEEE International Conference on Robotics and Automation*, pages 2992–2997, May 2001.

[19] L. Chaimowicza, V. Kumar, and M. Camposa. A framework for coordinating multiple robots in cooperative manipulation tasks. In G.T. McKee and P.S. Schenker, editors, *Proc. SPIE 4571 - Sensor Fusion and Decentralized Control in Robotic Systems IV*, pages 331–336, 2001.

[20] Y.J. Chan and K.C. Yow. A strategy-driven framework for multi-robot cooperation system. *Control, Automation, Robotics and Vision, 2006. ICARCV '06. 9th International Conference on*, pages 1–6, December 2006.

[21] I. Chebbi, S. Tata, and S. Dustdar. The view-based approach to dynamic inter-organizational work-flow cooperation. Technical Report TUV-1841-2004-23, Vienna University of Technology, Salzburg, Austria, 2004.

[22] N.Y. Chong, T. Kotoku, K. Ohba, K. Komoriya, N. Matsushira, and K. Tanie. Remote coordinated controls in multiple telerobot cooperation. In *Proceedings of the 2000 IEEE International Conference on Robotics and Automation*, pages 3138–3143, April 2000.

[23] S. Christensen and L. Petrucci. Modular analysis of petri nets. *The Computer Journal*, 43(3):224–242, 2000.

[24] Lehmann D. and Rabin M. On the advantages of free choice: A symmetric and fully distributed solution to the dining philosophers problem. In *Principles Of Programming Languages*, pages 133–138, 1981.

[25] R. David and H. Alla. *Discrete, Continuous, and Hybrid Petri Nets*. Springer-Verlag, 2005.

[26] B.P. Gerkey and M.J. Matari. Multi-robot task allocation: Analyzing the complexity and optimality of key architectures. In *IEEE Int. Conf. on Robotics and Automation*, pages 3862–3868, September 2003.

[27] L. Gomes and J. Barros. Pnml-based composition in non-autonomous petri net models. In *Industrial Electronics, 35th Annual Conference of IEEE*, pages 4377–4382, February 2009.

[28] D. Grigori, F. Charoy, and C. Godart. Coo-flow: A process technology to support cooperative processes. *International Journal of Software Engineering and Knowledge Engineering*, 14(3), 2003.

[29] D. Grigori, H. Skaf-Molli, and F. Charoy. Adding flexibility in a cooperative work-flow execution engine. In *Proc. of the 8th Int. Conf. on High-Performance Computing and Networking*, pages 227–236, 2000.

[30] T. GubaÅa, M. Bubak, M. Malawski, and K. Rycerz. Semantic-based grid workflow composition. *Lecture Notes in Computer Science, Parallel Processing and Applied Mathematics*, 3911:651–658, 2006.

[31] J.E. Haddad and S. Haddad. Self-stabilizing scheduling algorithm for cooperating robots. In *Proc. ACS/IEEE Int. Conf. on Computer Systems and Applications*, 2003.

[32] A. Herzig and D. Longin. A logic of intention with cooperation principles and with assertive speech acts as communication primitives. In *Proceedings of the AAMAS'02*, pages 920–972, July 2002.

[33] E. Kindler, A. Martens, and W. Reisig. Inter-operability of workflow applications: Local criteria for global soundness. *Lecture Notes in Computer Science, Business process management*, 1806:235–253, 2000.

[34] K. Knorr. Dynamic access control through petri net work-flows. In *ACSAC-00: Proceedings of the 16th annual Computer Security Applications Conference*, page 159, Washington DC, USA, 2000. IEEE Computer Society.

[35] Y. Kotb and E. Baderdin. Synchronization among activities in a work-flow using extended work-flow petri nets. In *Proc. of the 7th IEEE Int. Conf. on E-Commerce Technology*, pages 548–551, 2005.

[36] Y. Kotb, S. Beauchemin, and J. Barron. Petri net-based cooperation in multi-agent systems. In *CRV*, pages 123–130, May 2007.

[37] Y.T. Kotb and A.S. Baumgart. An extended petri net for modeling workflow with critical sections. In *IEEE Conference on E-business Engineering, Beijing*, June 2005.

[38] Y.T. Kotb and A.S. Baumgart. Multi-level workflow modeling using extended workflow petri nets. In *26th International Conference On Application and Theory of Petri Nets and Other Models of Concurrency*, June 2005.

[39] P. Lima, H. Gracio, V. Veiga, and A. Karlsson. Petri nets for modeling and coordination of robotic tasks. In *Proc. of the IEEE Int. Conf. on Systems, Man, and Cybernetics*, pages 190–195, San Diego, USA, October 1998.

[40] C. Lin, K. Song, and G.T. Anderson. Agent-based robot control design for multi-robot cooperation. In *Proceedings of the 2005 IEEE International Conference on systems, Man and Cybernetics*, pages 542– 547, Oct 2005.

[41] J. Liu and J. Wu. *Multi-Agent Robotic Systems*. The CRC international Series on Computational Intelligence, 2001.

[42] E.J. Macias and M.P. de la Parte. Simulation and optimization of logistic and production systems using discrete and continuous petri nets. *Simulation*, 80:143–152, 2004.

[43] M. Mecella, B. Pernici, M. Rossi, and A. Testi. A repository of work-flow components for cooperative e-applications. In *Proc. IFIP TC8 Working Conf. on ECommerce/EBusiness*, 2001.

[44] S. Nepal, A. Fekete, P. Greenfield, J. Jang, D. Kuo, and T. Shi. A service-oriented work-flow language for robust interacting applications. In *Proc. Int. Conf. on Cooperative Information Systems*, pages 40–58, Agia Napa, Cyprus, November 2005.

[45] H. Noborio and M. Edashige. A cooperative path-planning for multiple automata by dynamic/static conversion. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, pages 1955–1962, July 1993.

[46] J. Park. Structural analysis and control of resource allocation systems using petri nets, 2000.

[47] M. Pedersen. Compositional definitions of minimal flows in petri nets. In *Proceedings of the 6th International Conference on Computational Methods in Systems Biology*, pages 288–307. Springer-Verlag, 2008.

[48] X. Pengcheng and P. Mengchu. A petri net siphon based solution to protocol-level service composition mismatches. In *Web Services, IEEE International Conference on*, pages 952–958, July 2009.

[49] M. Pesic and W. van der Aalst. Modeling work distribution mechanisms using colored petri-nets. *International Journal on Software Tools for Technology Transfer*, 9(3-4):327–352, 2007.

[50] M.K. Purvis, M.A. Purvis, and S. Lemalu. An adaptive distributed work-flow system framework. In *7th Asia-Pacific Software Engineering Conf.*, pages 311–318, Los Alamitos, CA, 2000. IEEE Computer Society Press.

[51] H. Reijers. *Design and Control of Workflow Processes: Business Process Management for the Service Industry. (Lecture Notes in Computer Science).* springer, 2002.

[52] P. Renaud, E. Cervera, and P. Martiner. Towards a reliable vision-based mobile robot formation control. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, pages 3176–3181, October 2004.

[53] L. Sauro and W. van der Hoek. Reasoning about action and cooperation. In *proceedings of the AAMAS'06*, pages 185–192, May 2006.

[54] W. van der Aalst. Verification of workflow nets. In *Proceedings of the 18th International Conference on Application and Theory of Petri Nets*, pages 407–426, 1997.

[55] W. van der Aalst. Interorganizational workflows: An approach based on message sequence charts and petri nets. *Systems Science*, 34(3):335–367, 1999.

[56] W. van der Aalst, V. Hee, and G. Houben. Modelling and analysing workflow using a petri-net based approach. In *In proceeding of the second workflow on computer support cooperative work, petri nets and related formalisms*, pages 31–50, June 1994.

[57] W. van der Aalst, N. Lohmann, P. Massuthe, C. Stahl, and K. Wolf. From public views to private views correctness-by-design for services. *Lecture Notes in Computer Science, Web Services and Formal Methods*, 4937:139–153, 2008.

[58] W.M. van der Aalst. Challenges in business process management: Verification of business processes using petri nets.

[59] W.M. van der Aalst. Workflow verification: Finding control-flow errors using petri-net-based techniques. *Lecture Notes in Computer Science*, 1804, 2000.

[60] W.M. van der Aalst and K. van Hee. *Workflow Management. Models, Methods, and Systems.* MIT Press, Cambridge, MA, USA, 2002.

[61] Dijkstra W. Hierarchical ordering of sequential processes. In *Acta Informatica 1(2)*, pages 115–138, 1971.

[62] T. Wei, F. Yushun, Z. MengChu, and Z. MengChu. A petri net-based method for compatibility analysis and composition of web services in business process execution language. *Automation Science and Engineering, IEEE Transactions on*, 6(1):94–106, January 2009.

[63] T. Wei, F. Yushun, Z. MengChu, and T. Zhong. Data-driven service composition in enterprise soa solutions: A petri net approach. *Automation Science and Engineering, IEEE Transactions on*, 7(3):686–694, July 2010.

[64] S. Yan and F. Wang. A cooperative framework for inter-organizational work-flow systems. In *27th Annual Int. Computer Software and Applications Conf.*, pages 64–71, November 2003.

[65] X. Yang and H. Han. A petri net-based model for aspect-oriented web service composition. In *Management and Service Science, International Conference on*, pages 1–4, October 2009.

[66] D. Yingying, H. Yan, and J. Jiangping. Introducing personality into the multi-robot cooperation. In *Proceedings of the 2005 American Control Conference*, pages 5003– 5008, June 2005.

[67] W. Zhang. Representation of assembly and automatic robot planning by petri net. *IEEE Trans. on Systems, Man, and Cybernetics*, 19(2):418–422, 1989.

[68] Z. Zhao, A. Belloum, C. Laat, A. Wibisono, and B. Hertzberger. Using jade agent framework to prototype an e-science workflow bus. In *Proceedings of Seventh IEEE International Symposium on Cluster Computing and the Grid (CCGRID07)*, pages 655–660. IEEE Computer Society, 2007.

[69] Z. Zhao, A. Belloum, A. Wibisono, F. Terpstra, P. Boer, P. Sloot, and B. Hertzberger. Scientific workflow management: between generality and applicability. In *Proceedings of Fifth International Conference on Quality Software (QSIC'05)*, pages 357–364. IEEE Computer Society, 2005.

Curriculum Vitae

Name:	Yehia Kotb
DATE OF BIRTH	December 22,1973
PLACE OF BIRTH	Alexandria, Egypt
Degrees:	2005 - 2011 doctor of philosophy.
	University of Western Ontario
	London, ON
	2005 - 2011 doctor of philosophy.
Awards:	Best Paper award CRV-2007
Related Work	Teaching Assistant 2005-2009